FACULTAD DE INFORMÁTICA | UNIVERSIDAD NACIONAL DE LA PLATA

TESINA DE LICENCIATURA

Título: Uso de códigos bidimensionales y posicionamiento para el reencuentro de mascotas con sus dueños.
Autores: Santiago Ribero Vairo
Director: Dra. Cecilia Challiol
Codirector: Dra. Silvia Gordillo
Carrera: Licenciatura en Sistemas

Resumen

En aquellos hogares en los cuales las mascotas son consideradas parte de la familia, el extravío de una mascota representa una situación angustiante tanto para el animal como para las personas. La prevención habitual consiste en colocarle una identificación a la mascota. Pero ocurrido el extravío, de no haberse tomado dicha precaución, el procedimiento habitual consiste en anunciar la búsqueda en la vía pública, junto a las características de la mascota y en algunos casos una recompensa. Esta práctica se ha replicado en los últimos tiempos a las redes sociales.
La diversidad de tecnologías de posicionamiento, los códigos bidimensionales y los servicios de notificación pueden ser combinados para ofrecer, desde la tecnología, una solución a la problemática. Las soluciones actuales son todas creadas ad-hoc. Esta es la principal motivación de esta tesis, donde se propone una solución de modelado para este tipo de problemática. En esta tesina se describe un modelo de solución que promueve y busca acelerar la recuperación de mascotas al generarles, por ejemplo, un código QR que puede ser leído por cualquier persona que cuente con un smartphone con tecnología de posicionamiento (por ejemplo GPS) para, a través de una serie de alertas automáticas, darle a conocer al dueño la posición de la mascota extraviada y facilitar el reencuentro con la misma. En base al modelo propuesto, se desarrolló un prototipo funcional con la funcionalidad mencionada.

Palabras Claves

Sistema para mascotas; Posicionamiento; GPS; Códigos QR; Aplicación Web basada en posicionamiento.

Trabajos Realizados

Se realizó un relevamiento bibliográfico de tecnologías afines con la problemática a resolver (códigos bidimensionales y posicionamiento).

También se realizó un relevamiento de sistemas sobre temáticas similares a la recuperación de mascotas, de los cuales se identificaron ventajas y desventajas de cada uno de ellos.

Se diseñó un modelo para dar solución a la problemática. Y en base es este modelo, se desarrolló un prototipo funcional (aplicación web responsive) para mostrar cómo un sistema de este tipo puede ponerse en práctica.

Conclusiones

El modelo propuesto es extensible, permitiendo así agregar nuevas funcionalidades en un futuro.

El análisis bibliográfico permitió determinar que existen diferentes tecnologías que se pueden usar para implementar el modelo propuesto. Por masividad, se eligió para el prototipo usar códigos QR para la identificación de mascotas. Estos se generan a través del prototipo (cada dueño puede imprimirlos las veces que necesite). El posicionamiento de los usuarios se realiza mediante la API de Posicionamiento de HTML5.

Trabajos Futuros

- *Enriquecer el Sistema con la funcionalidad para que los usuarios se comuniquen a través de él, es decir, que no sean visibles números telefónicos.*
- *Permitir seleccionar los mecanismos de notificación por los cuales el usuario quiere recibir los alertas, como así también el radio de movilidad que mascota tiene permitido andar (ej. una cuadra de su casa).*
- *Desarrollar la funcionalidad para lograr una comunicación rápida y efectiva entre personas con recursos y mascotas con necesidades a cubrir.*
- *Identificación de mascotas por fotos (imagen), y no solo usando códigos bidimensionales.*

Fecha de la presentación: Diciembre 2016

Índice General

Capítulo 1. Introducción. 3
1.1. Motivación 3
1.2. Objetivos 5
1.3. Estructura de la tesina 6
Capítulo 2. Background 7
2.1. Códigos bidimensionales 7
- QR-Code 9
- Códigos Aztec 12
- High Capacity Color Barcode (HCCB) 14
- PDF417 18
- DataMatrix 18
- MaxiCode 19

2.2. Mecanismos de posicionamiento para dispositivos móviles 20
- GPS 20
- A-GPS 22
- GLONASS 22
- BeiDou 24
- Wi-Fi positioning (WPS) 25

2.3. Sistemas de seguimiento de mascotas 27
- Cat@Log 27
- Punetha-Mehta 29
- FindingRover 29
- ScanAPet 30
- PerrosQR 31
- Registro e identificación de perros callejeros de Berazategui 32

Capítulo 3. Modelo propuesto 34
3.1. Caracterización de la problemática a resolver 34
3.2. Descripción del Modelo Propuesto. 37
Capítulo 4. Prototipo desarrollado 58

4.1. Estructura de la solución desarrollada. 58
4.2. Dificultades y decisiones de implementación 62
Capítulo 5. Casos de prueba del prototipo desarrollado 64
Capítulo 6. Conclusiones y Trabajos Futuros. 78
Referencias bibliográficas. 83

Capítulo 1. Introducción.

1.1. Motivación

Enfrentar el extravío de una mascota puede contarse entre las experiencias más angustiantes que puede padecer una persona a lo largo de su vida, desde el momento en que (como demuestran distintos artículos periodísticos[1][2][3]) es habitual que las mascotas sean consideradas miembros de la familia.

Por otra parte, es frecuente atestiguar en la vía pública la angustia de una mascota extraviada, que no cuenta más que con su instinto para volver a su hogar. Y la primera reacción de las personas ante esta situación es determinar si la mascota lleva collar o no, en búsqueda de una chapa de identificación.

En zonas urbanas no hace falta transitar muchas cuadras para encontrar en postes, vidrieras o paredes, anuncios notificando el extravío de alguna mascota, con su fotografía y descripción, con la esperanza de que quien accidentalmente haya visto al animal, y accidentalmente haya visto el anuncio, oficie de puente entre mascota y dueño, en algunos casos bajo el incentivo de una recompensa económica que para el dueño nunca será suficiente, pero estará limitada por su situación económica particular.

Bajo la misma dinámica, pero trasladada al ámbito virtual, es habitual encontrar en las redes sociales publicaciones con las características antes mencionadas, e incluso grupos dedicados exclusivamente a reunir estos anuncios, con miras a lograr un punto de encuentro entre dueños y personas solidarias que voluntariamente notifiquen la aparición de las mascotas.

Aproximándonos un poco más al aprovechamiento de la tecnología, existen sitios que, con fines comerciales, facilitan la recuperación de mascotas extraviadas. Algunos ofrecen la confección y envío de chapitas de identificación con códigos bidimensionales (como QR[4]) asociados a fichas de mascotas; otros hacen uso del reconocimiento facial; y otros permiten navegar perfiles de mascotas extraviadas. Algunos ejemplos de ello son *FindingRover*[5] (aplicación móvil con reconocimiento facial de las mascotas); *ScanAPet*[6] (empresa dedicada a la fabricación de chapas identificatorias que incluyen un código QR con un hipervínculo hacia una ficha creada por el

[1] http://www.infobae.com/2011/09/16/605985-argentina-el-pais-la-region-mas-mascotas-habitante/ (Último acceso: 18-ago-2016)
[2] http://entremujeres.clarin.com/hogar-y-familia/mascotas-familia-miembro-perros-gatos-pet_friendly-amor-hijos_0_1334867971.html (Último acceso: 18-ago-2016)
[3] http://www.pagina12.com.ar/diario/sociedad/3-298127-2016-04-29.html (Último acceso: 18-ago-2016)
[4] Más información de un generador de códigos QR se puede encontrar en: http://www.codigos-qr.com (Último acceso: 14-jul-2016)
[5] Página de *FindingRover*: http://www.findingrover.com (Último acceso: 14-jul-2016)
[6] Página de *ScanAPet:* http://scanapet.com/ (Último acceso: 3-ago-2016)

usuario para la mascota); *PerrosQR*[7] (empresa dedicada a la fabricación de chapas de identificación que incluyen un código QR que codifica un hipervínculo hacia una ficha con información de la mascota a elección del cliente, como por ejemplo información de contacto); *Back2Gether*[8] (aplicación para celulares que incluye un mapa y permite marcar la zona donde se extravió -o dónde apareció- el animal y dejar datos de contacto).

Sin embargo, ninguno de los ejemplos mencionados anteriormente hace un aprovechamiento considerable de la tecnología, y de la popularidad y características actuales de los dispositivos móviles (cámara, geolocalización, internet, teléfono). Estos sistemas tampoco cuentan con un mecanismo automático de notificación en tiempo real al dueño de la mascota, cuando la misma es encontrada por alguna persona.

Otro aspecto que no está presente en las aplicaciones mencionadas, es un modelo de solución para las mismas. Es decir, las aplicaciones son implementaciones puntuales pero no proponen un modelo flexible que pueda ir escalando para incorporar nuevos requerimientos que le permitan evolucionar en el tiempo.

Como ejemplo de lo anterior, ninguno de estos sitios aprovecha su plataforma para abrir un canal de contacto entre mascotas padeciendo *necesidades*[9] y posibles voluntarios. Muchas veces nos encontramos con disponibilidad de recursos que sabemos podrían ser de mucha utilidad para mascotas y animales en general. Entre estos cito como ejemplos los recursos materiales (una lona, una chapa, un abrigo, un collar, un colchón), económicos (dinero, vouchers), profesionales (es el caso de abogados, veterinarios), logísticos (camionetas, refugio temporal), alimenticios (desde bolsas de alimento balanceado hasta restos de comida), como así también medicamentos, una familia adoptiva, o simplemente tiempo de voluntariado. Al no contarse con un mecanismo de vinculación entre recursos de personas y necesidades de mascotas, muchos de estos recursos se desperdician, pudiendo haber sido aprovechados para construir refugios (recursos materiales), pagar tratamientos (recursos económicos), asistir médicamente a un animal (recursos profesionales), trasladar a una mascota minusválida (recursos logísticos), alimentar animales (recursos alimenticios), darle uso a la medicación de una mascota fallecida, adoptar a una mascota en lugar de comprarla, o dedicar tiempo a cuidar y tratar animales de un refugio.

La motivación principal de esta tesina es proponer una solución de modelado al problema de recuperación de mascotas extraviadas, basada en el posicionamiento de las mismas para facilitar la reunión con sus familias. La existencia de un modelo con estas características facilitará la construcción de sistemas que le permitan al dueño conocer la ubicación de una mascota extraviada. En esta tesis se propondrá un prototipo en base a dicho modelo.

---

[7] Página de *PerrosQR:* http://www.perrosqr.com (Último acceso: 14-jul-2016)

[8] Página de *Back2Gether: https://itunes.apple.com/ar/app/back2gether/id666053130?mt=8* (Último acceso: 3-ago-2016)

[9] Entendemos en esta tesis *necesidades* como aquellos recursos que puede necesitar una mascota, por ejemplo, un medicamento.

Respecto a la solución al mencionado problema de las necesidades (recursos) y voluntarios, a nivel de modelado solo nos focalizaremos en representar aquellos conceptos relevantes, sin entrar en detalle en esta tesis de aspectos relacionados con dicha funcionalidad (estos se mencionan en el Capítulo 6). En futuras extensiones del modelo propuesto se podría extender los conceptos presentados para contar, por ejemplo, con la funcionalidad de red social de voluntarios para reunir necesidades de mascotas con recursos de voluntarios. Este tipo de extensiones serán posibles gracias a que el modelo será planteado de manera flexible para ser extendido.

1.2. Objetivos

El objetivo principal de esta tesina es proponer una solución de modelado que contemple una representación posicional de las mascotas (determinada por la ubicación de dueños y extraños al momento de identificar al animal), como así también sus estados de extravío que, complementados con un mecanismo de notificaciones automáticas, conforme una solución robusta a la problemática de reubicación de mascotas extraviadas. Con este objetivo, se diseñara un modelo que contemple principalmente los siguientes eventos:

- *Notificación del extravío de la mascota*: este evento será disparado por el dueño de la misma, al reportarla como extraviada. Se registrará como posición de extravío la posición del dueño al momento de dispararse el evento.
- *Notificación del hallazgo de la mascota*: este evento será disparado automáticamente ante la identificación por parte de extraños de una mascota en el sistema (por ejemplo, una persona encuentra una mascota en la vía pública y lee su código QR). Se registrará como posición del hallazgo la posición en que la cual fue encontrada la mascota. Se notificará mediante una alerta automática al dueño que la misma ha sido encontrada.
- *Notificación de la recuperación de la mascota*: este evento será disparado por el dueño de la misma, al identificarla tras su extravío. Esto permite registrar que la mascota se reunió con el dueño, por ejemplo, cuando el dueño la fue a buscar al lugar donde la tenía la persona que la encontró. Cuando el sistema detecte un evento de este tipo deberá excluir de la lista de mascotas extraviadas a dicha mascota.

Por otro lado, se representarán los conceptos relacionados a las necesidades y recursos (estos se detallan en el Capítulo 6), pero no se entra en detalle en esta tesis de aspectos relacionados con la funcionalidad de reunir necesidades de mascotas con recursos de voluntarios. Esto podrá ser ampliado en futuros trabajos. En esta tesis solo se dejarán planteados los conceptos encontrados como una forma de indicar los mismos y permitiendo la extensibilidad del modelo propuesto.

Cabe destacar que el modelo se diseñará para que sea flexible y extensible, de manera tal de poder agregarle nuevas funcionalidades, y así cubrir otros aspectos no detallados en esta tesis.

Otro de los objetivos es implementar un prototipo funcional en base al modelo propuesto. Esto nos permitirá mostrar la viabilidad del diseño realizado. Dicho prototipo estará focalizado en implementar los eventos relacionados a las notificaciones automáticas mencionados anteriormente. Para la identificación de las mascotas el prototipo utilizará códigos QR mientras que para el posicionamiento de los usuarios del sistema se utilizará la API de posicionamiento de HTML5.

1.3. Estructura de la tesina

En este apartado se detalla el contenido de cada uno de los siguientes capítulos de la presente tesina.

En el Capítulo 2 se presenta un resumen del relevamiento bibliográfico y documental realizado sobre aspectos técnicos y funcionales relativos a la problemática planteada por esta tesina. Este relevamiento se realizó haciendo foco en tres aspectos: códigos bidimensionales, mecanismos de posicionamiento para dispositivos móviles, y sistemas de seguimiento de mascotas.

En el Capítulo 3 se describe el modelo propuesto, partiendo de la caracterización de la problemática, identificando los conceptos que la conforman, e incluyendo un diagrama de casos de uso que describe las funcionalidades que deberían dar solución a esta problemática. Luego, el capítulo describe el modelo propuesto, de manera incremental, partiendo de lo más general e incorporando gradualmente los detalles hasta llegar a cubrir todo el modelo, del cual se incluye un diagrama completo. Por último se incluyen diagramas de secuencia de las operaciones más relevantes del modelo para mostrar el funcionamiento del mismo.

En el Capítulo 4 se desarrolla una descripción técnica general del prototipo desarrollado, en la cual se detallan las herramientas utilizadas, los proyectos involucrados en la solución y sus relaciones, e incluye un apartado en el cual se describen diferentes dificultades técnicas que debieron superarse para lograr un prototipo con el alcance necesario para demostrar la viabilidad del modelo.

El Capítulo 5 está enfocado en presentar los resultados del desarrollo y las funcionalidades con las que logra dar solución a la problemática. Se incluyen capturas de pantalla de todas las vistas y formularios involucrados en la funcionalidad, mientras se describe la relación entre cada uno de esos componentes y el flujo de utilización por parte de dos usuarios: El dueño de una mascota, y un desconocido que la encuentra.

En el Capítulo 6 se presentan las conclusiones y se identifican algunos trabajos futuros surgidos del desarrollo de la presente tesina, tanto a nivel de modelado como para el prototipo.

Por último, se detalla la Bibliografía consultada a lo largo de la investigación, la cual es referenciada a lo largo de esta tesina.

Capítulo 2. Background

En este capítulo se presentarán, por un lado, distintos códigos bidimensionales desatancando las características principales de cada uno de ellos. Luego, se describirán distintos mecanismos de posicionamiento. Para finalizar se presentarán distintos sistemas existentes que permiten brindar algún tipo de servicio relacionado a la temática de las mascotas.

2.1. Códigos bidimensionales

Como indica [Subpratatsavee, 2014], los códigos bidimensionales son patrones geométricos de dos dimensiones. Los códigos de barras de dos dimensiones tienen más capacidad que los de una dimensión utilizando un espacio menor, ya que pueden almacenar información vertical y horizontalmente dando soporte al almacenamiento de gran cantidad de información sin necesidad de acceder a una base de datos o cualquier otra herramienta de almacenamiento. Generalmente, los códigos bidimensionales contienen pixeles negros sobre un fondo blanco.

Actualmente existen diversas implementaciones de códigos bidimensionales, con algunas ventajas y desventajas entre unas y otras, difiriendo en factores como:

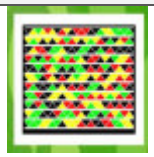
- *densidad de la información* (cuánta información, por ejemplo expresada en bits, cabe en un determinado área)
- *corrección de errores* (el código podría sufrir daños o deformaciones, pero algunas implementaciones cuentan con mecanismos de redundancia para hacer interpretable un código que no esté en óptimas condiciones)
- *derechos y patentes* (algunos formatos requieren licencia para su uso)
- *disponibilidad de aplicaciones de decodificación* (algunos formatos cuentan con muchas ventajas entre las categorías mencionadas, pero no cuentan con aplicaciones de decodificación que estén popularizadas).
- *popularidad* (algunos formatos ni siquiera son visualmente identificados como tales por el público en general).

La elección de la implementación de código bidimensional a utilizar dependerá de las características particulares del proyecto que se desarrolle. Por ejemplo, si el sistema consistiera en el posicionamiento de camiones, probablemente el factor de densidad de la información tendría poco peso, ya que la impresión del código bidimensional podría tener el tamaño que fuera necesario, al estar adherido a un camión. Un caso opuesto es el del posicionamiento de mascotas, en las que el tamaño del código físico es determinante, al estar adherido al collar. Otro ejemplo de estas consideraciones podría estar dado por el soporte de corrección de errores brindado por la implementación de código bidimensional seleccionada. Es decir, si la implementación contempla en sus algoritmos de codificación (generación) y decodificación (lectura) mecanismos (parametrizables o no) de redundancia de datos para hacerlo legible aún ante daños o deformaciones. A mayor redundancia, mayor corrección de errores, pero también mayor tamaño.

En el caso de los camiones, si el código estuviese adherido sobre la cara interna del parabrisas o luneta, la improbabilidad del daño que podría recibir el código haría que el factor de corrección de errores tuviera poco peso. En el caso del posicionamiento de mascotas, el código estaría en constante interacción física con el entorno, lo que le da relevancia al factor de corrección de errores a la hora de elegir la implementación a utilizar.

Actualmente, los códigos bidimensionales que son más usados son QR Code (*Quick Read Code*) [ISO/IEC 18004:2000, 2000], Aztec Code [ISO/IEC 24778:2008, 2008], PDF417 (*Portable Data File*) [ISO/IEC 15438:2015, 2015], DataMatrix [ISO/IEC 16022:2006, 2006] y MaxiCode [ISO/IEC 16023:2000, 2000]. Sin embargo, algunos códigos, como los HC2D (*High Capacity 2D*) [Subpratatsavee, 2014] tienen mayor capacidad pero no son tan usados. Las características y propiedades de los códigos bidimensionales son presentadas en la Tabla 2.1.

Tabla 2.1: Resumen comparativo de los códigos bidimensionales más usados [Subpratatsavee, 2014]

	QR Code	Aztec Code	HCCB	PDF417	Data Matrix	MaxiCode
Ejemplo de Imagen						
Capacidad (caracteres)	4296	3067	Indefinido	1848	2335	93
Alta capacidad	Si	Si	Si	Si	Si	No
Alta velocidad de lectura	Si	No	Si	No	No	Si
Pequeño	Si	No	Si	No	Si	No
Aplicaciones	Todas las industrias	Industrias de la aviación y el transporte	Contenidos audiovisuales	Oficina	Industria de la medicina	Importación y exportación de productos industriales
Licencia	Libre	Libre	Propietario	Libre	Libre	Libre

Se puede apreciar en la Tabla 2.1 que en una primera revisión de las características de los códigos, los QR poseen una mayor aplicabilidad, y tienen también una alta capacidad y alta velocidad de lectura. A continuación mencionaremos más detalles de cada uno de los códigos presentados. Se evaluarán características, ventajas, desventajas y usos frecuentes de diversos códigos bidimensionales existentes. Cabe destacar que en esta tesis se hará diferencia entre dos conceptos:

- *Código*: para referirnos a un formato particular, independientemente de su contenido (por ejemplo: código QR, código DataMatrix, códigos bidimensionales).
- *Símbolo*: para referirnos a una generación concreta de un código específico (por ejemplo, *un símbolo QR en un collar*, *un símbolo DataMatrix dañado*).

- **QR-Code**

Es un código bidimensional desarrollado en 1994 por *Denso Wave Corporation*. El origen de su nombre (*Quick Response Code*) se debe a que fue desarrollado para mejorar la velocidad de lectura de códigos bidimensionales de estructura compleja. En la Figura 2.1 se puede apreciar un ejemplo de este tipo de código.

Figura 2.1: Ejemplo de código QR.

Como indica [Bui, 2014], los QR son códigos bidimensionales con características muy buenas que los hacen ampliamente utilizados en la actualidad. Cuenta con seis valiosas características:

- alta capacidad de codificación,
- tamaño de impresión pequeño,
- capacidad de Kanji y Kana (*caracteres japoneses*),
- legibilidad pese a suciedad y daños (gracias a redundancia),
- legible en cualquier dirección en 360°,
- y característica de concatenación estructurada (un código QR se puede dividir en códigos QR más pequeños que, al concatenarlos, contienen la misma información que el código original).

Estas características mencionadas son alcanzadas debido a que posee las siguientes cinco propiedades especificadas en [Bui, 2014]:

- *Conjunto de caracteres codificables:*
  - Datos numéricos (dígitos del 0 al 9).
  - Datos alfanuméricos (dígitos del 0 al 9, letras A-Z mayúsculas y nueve caracteres más: *espacio $ % * + - . / :*)
  - Datos binarios.
  - Caracteres Kanji.

- *Tamaño de símbolo:*

  Existen 40 versiones de códigos QR, desde 1 hasta 40, donde el tamaño mínimo es 21x21 módulos (versión 1) y el máximo es 177x177 módulos (versión 40).

- *Caracteres por símbolo:*

  Para la máxima capacidad de los códigos QR versión 40 con nivel de corrección bajo, se pueden codificar:

  Datos numéricos: 7089 caracteres.
  Datos alfanuméricos: 4296 caracteres.
  Datos binarios: 2953 bytes.
  Caracteres Kanji: 1817 caracteres.

- *Nivel de corrección seleccionable:*

  Los códigos QR pueden incluir información para proveer la capacidad de corrección de errores. Gracias a esta capacidad es posible leer el código incluso si este presenta algunas distorsiones o daños. Existen cuatro niveles diferentes de corrección de errores, cada uno asociado a un porcentaje aproximado del área del símbolo que puede ser restaurada como máximo. Los cuatro niveles de corrección son:

  - L (Low): 7%
  - M (Medium): 15%
  - Q (Quartile): 25%
  - H (High): 30%

  El nivel de corrección seleccionado al momento de generar el código debe determinarse en función de la distorsión que pueda llegar a sufrir el símbolo por el entorno.

- *Estructura del símbolo:*

  En la Figura 2.2 se puede apreciar la estructura de un código QR Versión 2.

Figura 2.2 Estructura del código QR [Bui, 2014].

A continuación se detallan los componentes presentados en la Figura 2.2:

- *Margen*: es utilizado para aislar el código de otra información. Esta región tiene un ancho de cuatro módulos.
- *Patrón de detección de posición*: ubicado en tres de las cuatro esquinas. Permite la lectura de 360 grados (omnidireccional) del código a alta velocidad.
- *Patrón de alineación*: Este patrón permite al lector de código QR corregir la distorsión cuando el código es doblado o curvado. La cantidad de patrones de

alineación utilizada depende de cuánta información es codificada, apareciendo desde la versión 2 en adelante.

- *Información de formato*: Contiene el patrón de frecuencia de corrección de errores y la máscara de código QR (mecanismo para facilitar la lectura). La información de formato es la primera en leerse cuando el código es decodificado.
- *Información de versión*: Identifica la versión desde la 1 hasta la 40.
- *Área de datos*: es un arreglo de filas y columnas. Cada celda es almacenada como un número binario (1 y 0). Los códigos de corrección de errores también son insertados en esta región.
- *Corrección de errores*: se aplica para restaurar los datos cuando parte del código QR se pierde. Como se mencionó, el nivel de restauración varía entre cuatro niveles.
- *Sincronización*: Ayuda a detectar la posición de cada celda en el código QR.

En la industria se utilizan lectores de códigos con una resolución de lectura suficiente para interpretar todas las versiones de códigos QR. Pero para aplicaciones móviles se recomienda utilizar solo las versiones 1 a 10 (57 x 57) teniendo en cuenta las limitaciones de las cámaras de celulares, esto lo especifican los autores en [Kato & Tan, 2007]. Si bien esta recomendación data del año 2007, mediante pruebas actuales con una cámara de 12 megapíxeles pudo determinarse la vigencia de la misma.

Los códigos QR pueden ser fácilmente generados utilizando generadores online gratuitos [10]. Estos pueden ser impresos en papel utilizando impresoras comunes y pueden ser adheridos a cualquier objeto. Son habitualmente utilizados para almacenar URLs u otros identificadores pequeños como emails y números de teléfono que pueden ser leídos utilizando dispositivos móviles.

Se ha establecido como un estándar ISO que se ha definido en [ISO/IEC 18004:2000, 2000]. Los códigos QR son de uso libre y abierto desde que se da a conocer su especificación y los derechos de patente (propiedad de *Denso Wave Corporation*) no se ejercen.

- ### Códigos Aztec

En [Zhang, 2010] se define a Aztec como un código bidimensional creado por *Andrew Longacre Jr* en 1995. El código fue publicado por *AIM International* en 1997 y a pesar de que el código está patentado, ha sido liberado al dominio público. Los códigos Aztec fueron diseñados para ser fáciles de imprimir y de decodificar. Es utilizado en diversas áreas. En la Figura 2.3 se puede apreciar un ejemplo de este tipo de códigos.

[10] URLs de algunos generadores de códigos QR gratuitos:
http://goqr.me/ (Último acceso: 08-ago-2016)
http://www.qrcode.es/es/generador-qr-code/ (Último acceso: 08-ago-2016)
http://es.qr-code-generator.com/ (Último acceso: 08-ago-2016)

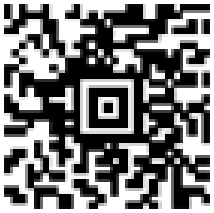


Figura 2.3: Ejemplo de código Aztec.

Los códigos Aztec poseen como ventajas: una lectura rápida y en todas las direcciones, alta capacidad de almacenamiento, alta fiabilidad y seguridad. Veamos más detalles de estas ventajas:

- Lectura super veloz. Es la característica clave que lo distingue de otros códigos bidimensionales como PDF417 y códigos QR.
- Los códigos Aztec no necesitan de un margen que delimite el símbolo, más allá de que algunos lectores lo requieran.
- Existen 32 tamaños con codificación de errores *Reed-Solomon* seleccionable por el usuario desde el 5% al 95% de la región de datos.
- Los códigos Aztec son robustos sobre diversas tecnologías de impresión. También son propicios para ser presentados en pantallas de dispositivos móviles y celulares.
- Todos los valores de 8 bits pueden ser codificados. Valores de 0 a 127 son interpretados como ASCII mientras que valores entre 128 y 255 son interpretados como ISO 8859-1, alfabeto latino n. 1.

Los símbolos son cuadrados sobre una grilla con una diana central. Las esquinas de la diana incluyen marcas de orientación permitiendo que el código sea legible si es rotado o reflejado. La decodificación comienza en las esquinas con 3 pixeles negros y continúa en sentido horario por las esquinas con dos, uno y cero pixeles negros.

Se ha establecido como un estándar ISO que se ha definido en [ISO/IEC 24778:2008, 2008]. En la Tabla 2.2 se pueden apreciar capacidades de los diferentes tipos de códigos Aztec.

Tabla 2.2: Capacidades de los diferentes tipos de códigos Aztec[11]

	Full	Compact	Rune
Ejemplo de Imagen			
Tamaño	19x19 – 151x151 módulos	15x15 – 27x27 módulos	11x11 módulos
Capacidad para texto (caracteres)	15 – 3067	12 – 89	0
Capacidad numérica (dígitos)	18 - 3832	13 - 110	3
Capacidad binaria (bytes)	8 - 1914	6 – 53	0

- ## High Capacity Color Barcode (HCCB)

Estos códigos fueron desarrollados dentro de *Microsoft Research*[12], cuenta entre sus ventajas una alta densidad de almacenamiento, posibilidad de uso de firma digital, la posibilidad de almacenar múltiples datos en un mismo código, y robustez con dispositivos móviles. Como indica [Parikh, 2008], los códigos HCCB pueden ser utilizados para una variedad de aplicaciones, pero su uso más común es la identificación unívoca de contenidos audiovisuales comerciales como videojuegos, películas, grabaciones digitales y otros recursos. En la Figura 2.4 se puede apreciar un ejemplo de este tipo de código.

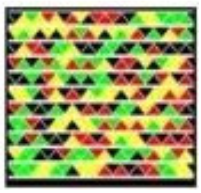


Figura 2.4: Ejemplo de código HCCB.

Como se puede apreciar en la Figura 2.4, los códigos HCCB contienen filas de símbolos (triángulos) de diferentes colores: negro, rojo, verde y amarillo. La cantidad de símbolos en una fila siempre es igual a un múltiplo entero de la cantidad de filas, el cuál puede variar. Están diseñados para estar enmarcados por una línea negra y un margen blanco. Este patrón fue

[11] http://help.accusoft.com/SAAS/pcc-for-acs/Aztec.html (Último acceso: 8-ago-2016)
[12] https://www.microsoft.com/en-us/research/project/high-capacity-color-barcodes-hccb/ (Último acceso: 6-ago-2016)

diseñado para funcionar como referencia visual para detectar el código en una imagen. El borde negro inferior es más grueso que el de los tres lados restantes, cumpliendo la función de orientar el código al poder estar ubicado en cualquier dirección. Los últimos 8 símbolos de la última fila siempre están en el orden fijo negro-rojo-verde-amarillo (2 triángulos consecutivos por color) y son utilizados para determinar la tonalidad con que aparece cada color en todo el código. Las filas están separadas por líneas blancas.

Esta tecnología toma ventaja de la capacidad que tienen los dispositivos móviles para procesar los colores, y del uso de formas triangulares en el código impreso para reducir el tamaño de los códigos. En la Figura 2.5 se puede apreciar un ejemplo de código de 8 colores almacenando 84 bytes, mientras en la Figura 2.6 se puede apreciar un ejemplo de código de 4 colores que almacena 58 bytes. En la Figura 2.7 se puede apreciar más nivel de detalle de estos códigos.

Figura 2.5: Ejemplo de código de 8 colores almacenando 84 bytes

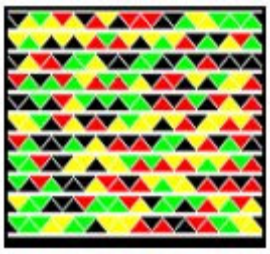


Figura 2.6: Ejemplo de código de 4 colores almacenando 58 bytes

Matriz monocromática almacenando 1 byte utiliza 8 símbolos

Código de 4 colores almacenando 1 byte utiliza 4 símbolos

Código de 8 colores almacenando 1 byte utiliza 2.66 símbolos

Figura 2.7: Detalles del tramado de los códigos HCCB.

En la Figura 2.8 se puede apreciar el almacenamiento de ítems individuales en un mismo símbolo, en el cual se describe un número, un mail y una dirección URL.

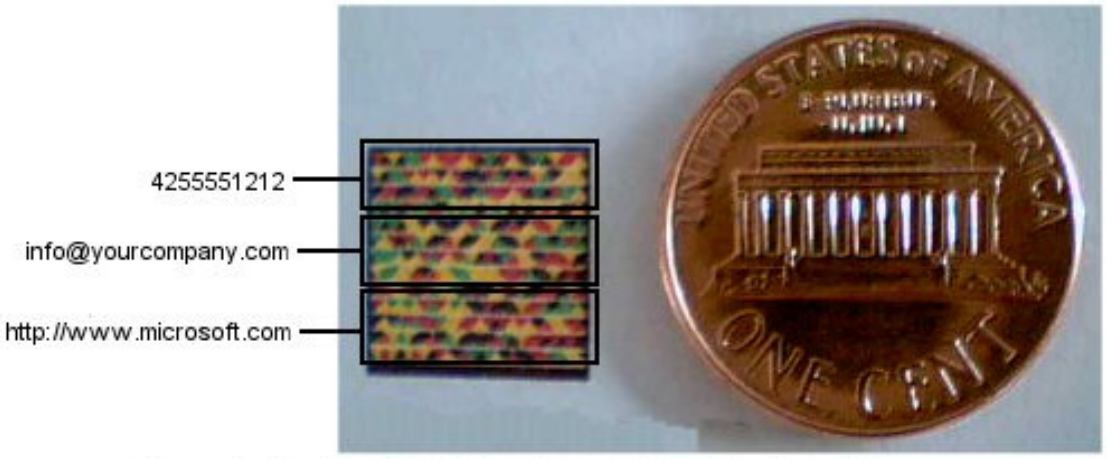


Figura 2.8: Almacenamiento de ítems individuales en un mismo código (A escala, leído por una webcam en resolución 320x240).

En la Figura 2.9 se puede apreciar la lectura de un código con diferentes niveles de luz y enfoque.

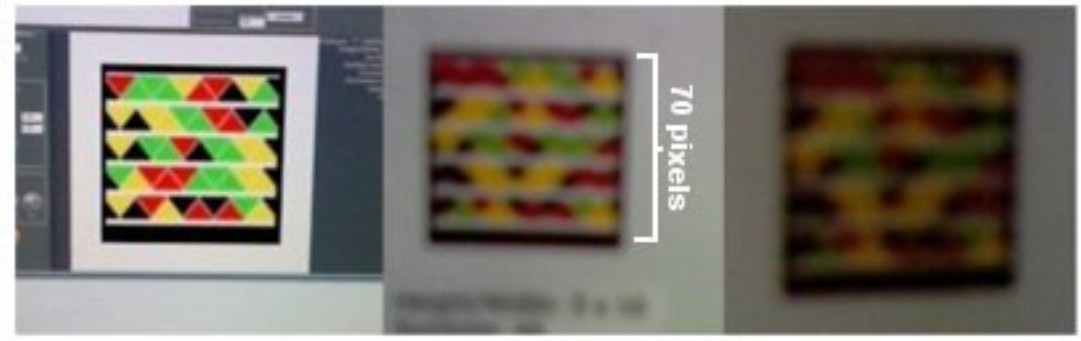
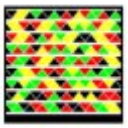


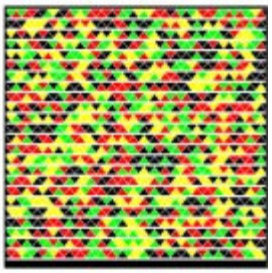
Figura 2.9: Ejemplos de lectura exitosa de un código de 50 símbolos a diferentes niveles de luz y enfoque (La longitud de lado del código capturado es de 70 pixeles).

En la Figura 2.10 se detallan tres códigos HCCB, uno de ellos sin firmar y los otros dos con firmas específicas, se puede apreciar que esto hace que varíe la cantidad de información que tiene cada uno de los códigos.



Sin firmar

Firmado utilizando firma Microsoft ECC[13]



Firmado utilizando firma RSA-1024[14]

Figura 2.10: Ejemplos de códigos sin firmar o con distintas firmas.

A continuación se listan ventajas del HCCB por sobre los QR Codes:

- Los códigos HCCB pueden ser firmados digitalmente ya sea con ECC como con RSA, y el lector deberá verificar la firma (si bien esto es técnicamente posible con cualquier código, los lectores actuales en el mercado de repositorios de aplicaciones no lo verifican).
- La especificación de HCCB fue diseñada para tomar ventaja del débil mecanismo de autofoco en las lentes de celulares.
- HCCB fue también diseñado para tomar ventaja de la variación de condiciones de luz.
- Debido a las formas triangulares de los símbolos HCCB, un código HCCB en blanco y negro puede contener la misma cantidad de datos en un espacio más reducido.
- Debido a las paletas alternativas de 4 y 8 colores, puede alcanzarse mayor compresión con los mismos datos en el código HCCB físico.
- HCCB puede almacenar diversidad de datos, tales como una dirección URL, texto y un vcard en un único código.

Respecto a las desventajas de los códigos HCCB respecto a los códigos QR se pueden mencionar:

- La licencia para imprimir códigos HCCB o desarrollar lectores para decodificar códigos es 100% propietaria.

[13] *Elliptic Curve Cryptography*, mecanismo de encriptación de Microsoft.
[14] *Rivest, Shamir y Adleman*, mecanismo de encriptación con clave de 1024 bits.

- *Microsoft Tag*, la implementación más popular de HCCB, requiere una conexión para acceder a los datos del código. Pueden verse los *Microsoft Tags* como URLs de redireccionamiento.
- Actualmente la única aplicación para decodificar códigos HCCB en los repositorios de aplicaciones más populares es *Microsoft Tag*, la cual está discontinuada.
- Las aplicaciones de decodificación de códigos más populares saben interpretar una gama de códigos que le permiten al usuario abstraerse del tipo concreto. El tipo de licencia de HCCB dificulta la incorporación de su decodificación en este tipo de aplicaciones genéricas, lo cual deriva en que el usuario deba identificar el tipo de código para determinar con qué aplicación decodificarlo.

- **PDF417**

Como indica [Zhang, 2008], los códigos PDF417 (*Portable Data File*) fueron desarrollados sobre la base de un código de barras convencional. Además de las características de un código de barras, tiene también otras ventajas, como una alta densidad de almacenamiento de información, potentes capacidades de corrección de errores y bajo costo. Es un método ideal para la transferencia de información, y es ampliamente utilizado en transferencias internacionales, distribución de materiales y servicio de correo, sin depender de conexión a redes y acceso a datos. Los códigos PDF417 pueden contener 1848 letras o 2792 caracteres numéricos. Puede comprimirse y decodificarse información como fotografías, sonidos y firmas en un PDF417.

Es uno de los códigos bidimensionales más maduros y extendidos. Es habitualmente descripto como un código de barras (de una dimensión) apilado. En la Figura 2.11 se muestra un ejemplo de este tipo de código.

Figura 2.11: Ejemplos de códigos PDF417.

- **DataMatrix**

En [Biao, 2007] se indica que cada símbolo DataMatrix consta de regiones de datos que contienen módulos nominalmente cuadrados establecidos en forma regular: los bordes izquierdo e inferior están conformados por una línea continua de bloques adyacentes que determinan dos bordes en forma de L. Estas líneas son utilizadas principalmente para determinar el tamaño físico, la orientación y la distorsión del símbolo. Los dos lados opuestos están conformados por la alternación de módulos claros y oscuros. Son utilizados principalmente para definir la estructura del módulo del símbolo, pero también pueden servir

de asistencia para determinar tamaño físico y distorsión. En la Figura 2.12 presenta un ejemplo de este tipo de código para clarificar así los detalles mencionados.

Figura 2.12: Ejemplo de símbolo DataMatrix.

Se puede apreciar en la Figura 2.12, que el código está conformado por módulos acomodados en un patrón cuadrado o rectangular. Posee patrones de redundancia para la recuperación de errores. En función al nivel de recuperación puede almacenar mayor o menor número de caracteres. Al mismo tiempo, la longitud de la información codificada depende de la dimensión del símbolo utilizado. Puede ser útil cuando es necesario un código de pequeño tamaño y cuadrado. En comparación con el código MicroQR[15], en el mismo tamaño y nivel de recuperación, DataMatrix puede contener mayor cantidad de información. Se ha establecido como un estándar ISO que se ha definido en [ISO/IEC 16022:2006, 2006].

- **MaxiCode**

Como se indica en [Borkowski, 1994], MaxiCode es una simbología bidimensional desarrollada por *United Parcel Service* como una alternativa de alta densidad a los códigos de barras. Un símbolo MaxiCode consiste de un arreglo de una pulgada conformado por hexágonos rodeando un patrón de ojo de buey (diana simétrica centrada). Más de 93 caracteres de información pueden ser almacenados en un mismo símbolo, el cual incorpora redundancia de datos. Y hasta 8 símbolos MaxiCode pueden encadenarse juntos para transportar más datos. En la Figura 2.13 se puede apreciar un ejemplo de este tipo de código.

Figura 2.13: Ejemplos de códigos MaxiCode.

La diana simétrica centrada es útil en la localización automática del símbolo independientemente de la orientación, y permite que los símbolos MaxiCode sean escaneados incluso en un paquete trasladándose a gran velocidad. Se ha establecido como un estándar ISO que se ha definido en [ISO/IEC 16023:2000, 2000].

[15] Versión reducida de QR, especificada en el mismo estándar [ISO/IEC 18004:2000, 2000]

2.2. Mecanismos de posicionamiento para dispositivos móviles

En esta sección se describen los mecanismos de posicionamiento (geolocalización) utilizados por algunos de los dispositivos móviles más populares actualmente. En la Tabla 2.3 se puede apreciar un resumen de cada uno de estos mecanismos y qué dispositivos soporta cada uno de ellos. Luego, cada mecanismo se presentará con mayor nivel de detalle.

Tabla 2.3: Mecanismos de posicionamiento utilizados por dispositivos populares de gama media y alta[16].

	GPS	A-GPS	Glonass	BeiDou	Wi-Fi positioning
iPhone 6s	Si	Si	Si		Si
Samsung Galaxy S6	Si	Si	Si	Si	
Samsung Galaxy J5	Si	Si	Si	Si	
Motorola Moto G4	Si	Si			
iPhone 4s		Si	Si		
Nokia Lumia 1520	Si	Si	Si		Si
Nokia Lumia 930		Si	Si		Si
HTC 10	Si	Si	Si	Si	

A continuación se presentarán en detalles los aspectos más relevantes de los mecanismos mencionados en la Tabla 2.3.

- **GPS**

  Según información oficial del Gobierno de los Estados Unidos relativa al *Sistema de Posicionamiento Global*[17] (GPS) este es un sistema de radionavegación de los Estados Unidos de América, basado en el espacio, que proporciona servicios fiables de posicionamiento, navegación, y cronometría gratuita e ininterrumpidamente a usuarios civiles en todo el mundo. A todo el que cuente con un receptor del GPS, el sistema le proporcionará su posición y la hora exacta en cualesquiera condiciones atmosféricas, de día o de noche, en cualquier lugar del mundo y sin límite al número de usuarios simultáneos.

  El GPS se compone de tres elementos: los satélites en órbita alrededor de la Tierra, las estaciones terrestres de seguimiento y control, y los receptores del GPS propiedad de los usuarios. Desde el espacio, los satélites del GPS transmiten señales que reciben e identifican los receptores del GPS; ellos, a su vez, proporcionan por separado sus coordenadas tridimensionales de latitud, longitud y altitud, así como la hora local precisa.

[16] http://www.phonearena.com/ (Último acceso: 10-dic-2016)
[17] http://www.gps.gov/ (Último acceso: 10-dic-2016)

Hoy están al alcance de todos en el mercado los pequeños receptores del GPS portátiles. Con esos receptores, el usuario puede determinar con exactitud su posición y desplazarse fácilmente al lugar a donde desea trasladarse, ya sea andando, conduciendo, volando o navegando. El GPS es indispensable en todos los sistemas de transporte del mundo ya que sirve de apoyo a la navegación aérea, terrestre y marítima. Los servicios de emergencia y socorro en casos de desastre dependen del GPS para el posicionamiento y coordinación horaria de misiones para salvar vidas. Actividades cotidianas como operaciones bancarias, de telefonía móvil e incluso de las redes de distribución eléctrica, ganan en eficiencia gracias a la exactitud cronométrica que proporciona el GPS. Agricultores, topógrafos, geólogos e innumerables usuarios trabajan de forma más eficiente, segura, económica y precisa gracias a las señales accesibles y gratuitas del GPS.

El estado actual de los satélites de la constelación GPS es descripto en la Tabla 2.4. Se puede observar que esta información está actualizada al 10-12-2016, y la misma puede variar en el tiempo.

Tabla 2.4: Estado de la constelación GPS al 10-dic-2016[18]

Estado de la constelación GPS al 10-dic-2016	
Total de satélites	32
Operativos	31
Puesta en marcha	--
Mantenimiento	1
Desmantelamiento	--

En la Figura 2.14 se presenta el trazo de los satélites de GPS. Dicha gráfica es del 2013 tomada de [Xingxing Li et. al, 2014].

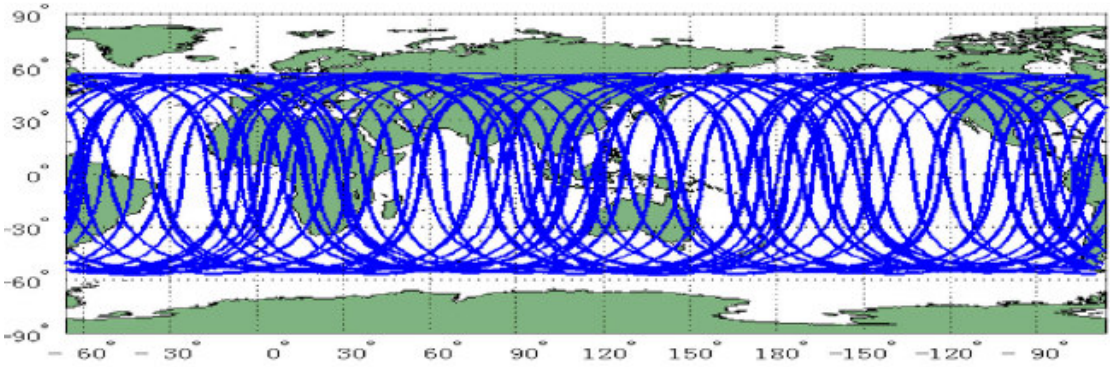


Figura 2.14: Trazo de satélites GPS a septiembre del 2013 [Xingxing Li et. al, 2014].

[18] https://www.glonass-iac.ru/en/GPS/ (Último acceso: 10-dic-2016)

- **A-GPS**

  La mayoría de los teléfonos celulares habilitados para GPS emplean una tecnología conocida como GPS Asistido (A-GPS) [Zandbergen, 2009]. Con A-GPS muchas de las funciones de un receptor GPS completo son realizadas por un servidor de posicionamiento GPS remoto (accedido a través de las torres de telefonía celular). Este servidor remoto proporciona al dispositivo móvil A-GPS información sobre la órbita de los satélites y el reloj; la posición inicial y la estimación del tiempo; y el cálculo de la posición. El dispositivo móvil contiene un receptor GPS muy básico que necesita sincronizarse con satélites dados que son visibles y transferir información de pseudo-rango al servidor de posicionamiento a través de la red celular. Con A-GPS el dispositivo móvil no necesita decodificar los mensajes GPS para cada satélite o realizar una búsqueda extensa de satélites visibles cuando el sistema está encendido. Esto se traduce en un consumo de energía reducido y en un tiempo de inicialización considerablemente menor. La mayoría de los proveedores de servicios celulares han adoptado A-GPS como la tecnología de elección para cumplir con los requisitos de la *Comisión Federal de Comunicaciones de los Estados Unidos* (FCC, por sus siglas en inglés) para obtener información sobre los servicios de posicionamiento. La FCC requiere que los sistemas de telefonía móvil ubiquen a la persona que llama en un radio de 50 metros para el 67% de las llamadas y 150 metros para el 95% de las llamadas. Sin embargo, el GPS y el A-GPS no funcionan muy bien en áreas urbanas de alta densidad debido a la limitada visibilidad de los satélites y normalmente no funcionan en interiores debido a obstrucciones de señal. Como resultado, la disponibilidad de posicionamiento confiable es limitada en áreas donde la gente pasa la mayor parte de su tiempo [Zandbergen, 2009].

  Se han desarrollado una serie de sistemas de posicionamiento en interiores para superar las limitaciones del A-GPS. Se basan en señales terrestres y utilizan señales de red celular, señales WiFi, Bluetooth, infrarrojos, ultrasonidos u otras frecuencias de radio. El posicionamiento por WiFi se detallará posteriormente.

- **GLONASS**

  GLObal'naya Navigatsionnay Sputnikovaya Sistema es un *Sistema Global de Navegación por Satélite* (GNSS) [Honkova, 2013] desarrollado inicialmente por la Unión Soviética, que se basa en una constelación de satélites activos, patrocinados por el *Ministerio de Defensa de la Federación de Rusia*, que transmiten continuamente señales codificadas en dos bandas de frecuencias que pueden recibir los usuarios en cualquier lugar de la superficie de la Tierra para identificar su posición y velocidad en tiempo real. La aplicación primaria de GLONASS es la transmisión de posición y tiempo[19].

[19] https://ilrs.cddis.eosdis.nasa.gov/missions/satellite_missions/current_missions/g129_general.html (Último acceso: 12-dic-2016)

El sistema es una contrapartida del *Sistema de Posicionamiento Global* (GPS) de los Estados Unidos y ambos sistemas comparten los mismos principios en los métodos de transmisión y posicionamiento de datos. Está completamente operativo desde 1995.

La constelación actual (GLONASS-M) está conformada por 24 satélites en tres planos orbitales (8 satélites por plano) sin contar los de repuesto y los de fase de prueba. Cada uno tiene una vida útil de 7 años. Cubren una órbita circular a 19140 km de altitud, y su período orbital es de 11 horas, 15 minutos y 44 segundos[20]. Su precisión es habitualmente comparada con la de GPS. En este momento, la precisión del sistema es inferior a los 2.8 metros. Se espera que para el año 2020 se haya logrado una precisión de 0.6 metros [Honkova, 2013]. El estado actual de los satélites de la constelación GLONASS es descripto en la Tabla 2.5. Se puede observar que esta información está actualizada al 10-12-2016, y la misma puede variar en el tiempo.

Tabla 2.5: Estado de la constelación Glonass al 10-dic-2016[21]

Estado de la constelación Glonass al 10-dic-2016	
Total de satélites	27
Operativos	23
Puesta en marcha	--
Mantenimiento	1
Repuesto	2
En fase de pruebas de vuelo	1

En la Figura 2.15 se presenta la cobertura actual de los satélites de GLONASS. Lo que indica la referencia inferior de colores es la cantidad de satélites sobre una determinada región.

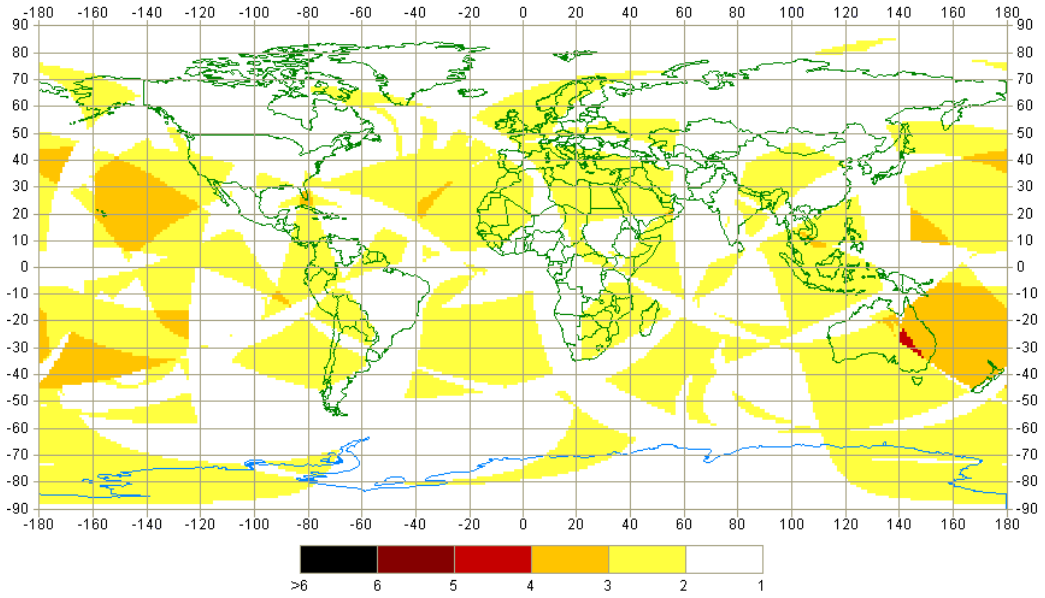


Figura 2.15: Cobertura de los satélites de GLONASS al 10-dic-2016[22]**.**

[20] https://www.glonass-iac.ru/en/guide/ (Último acceso: 12-dic-2016)
[21] https://www.glonass-iac.ru (Último acceso: 10-dic-2016)

En la Figura 2.16 se presenta el trazo de los satélites de GLONASS. Dicha grafica es del 2013 tomada de [Xingxing Li et. al, 2014].

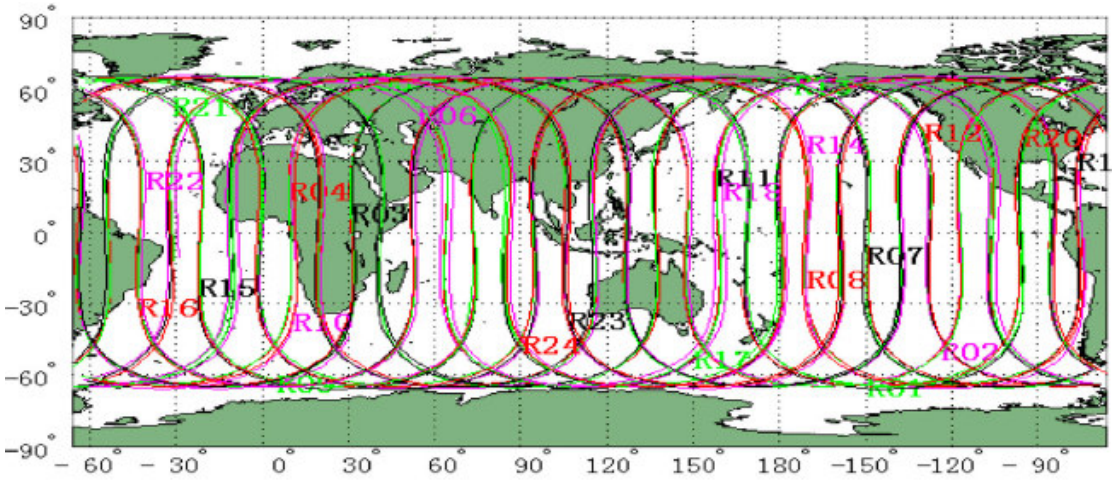


Figura 2.16: Trazo de satélites GLONASS a septiembre del 2013 [Xingxing Li et. al, 2014].

- **BeiDou**

BeiDou[23] es el sistema global de navegación por satélite de China que se ha desarrollado de forma independiente desde el punto de vista internacional.

El sistema de navegación por satélite *BeiDou* se compone de tres partes: la sección de espacio, la sección de tierra y la sección de usuario. La sección espacial contiene satélites de órbita geoestacionaria y satélites de órbita no geoestacionaria. La sección de tierra consiste en un cierto número de estaciones: incluyendo las estaciones de control principales, las estaciones de inyección y las estaciones de monitoreo. Y la sección de usuario incluye terminales del sistema *BeiDou*, algunos de ellos compatibles con otros sistemas de navegación por satélite.

Actualmente el sistema tiene cobertura sobre Asia y el Pacífico, pero está planificada la cobertura global para el año 2020. El sistema de navegación por satélite *BeiDou* proporcionará cobertura global con servicios de posicionamiento, navegación y sincronización, incluyendo dos tipos de modos de servicio: un servicio abierto y un servicio autorizado. El servicio abierto suministra sin cargo posición, velocidad y tiempo, con precisión de posicionamiento de 10 metros, precisión de la velocidad de 0,2 metros / segundo y precisión de tiempo de 10 nanosegundos. El servicio autorizado proporciona en forma más precisa la posición, la velocidad, el tiempo y los servicios de comunicaciones, así como un mayor nivel de integridad.

[22] https://www.glonass-iac.ru
[23] http://en.beidou.gov.cn/ (Último acceso: 10-dic-2016)

El estado actual de los satélites de la constelación *BeiDou* es descripto en la Tabla 2.6. Se puede observar que esta información está actualizada al 10-12-2016 y la misma puede variar en el tiempo.

Tabla 2.6: Estado de la constelación BeiDou al 10-dic-2016[24]

Estado de la constelación BeiDou al 10-dic-2016	
Total de satélites	18
Operativos	15
Puesta en marcha	1
Mantenimiento	--
Desmantelamiento	2
Repuesto	--

En la Figura 2.17 se presenta el trazo de los satélites de *BeiDou*. Dicha gráfica es del 2013 tomada de [Xingxing Li et. al, 2014].

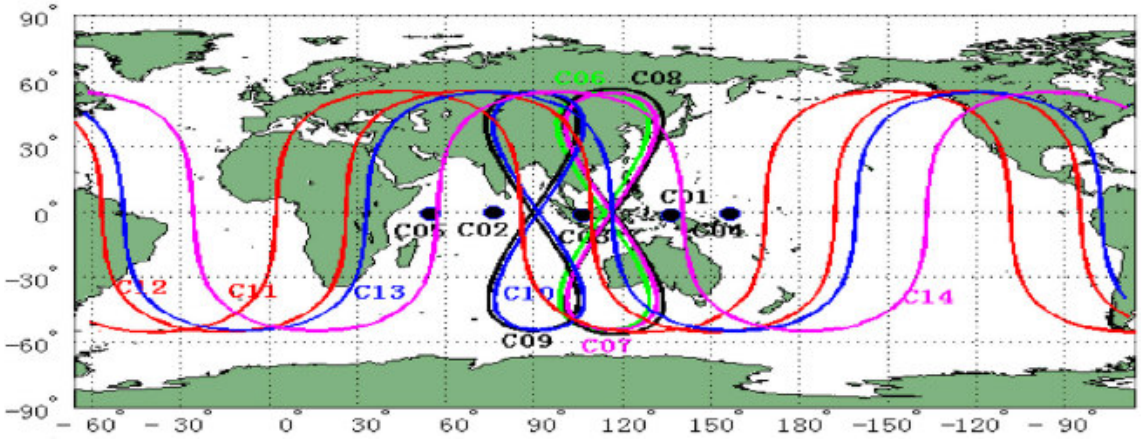


Figura 2.17: Trazo de satélites BeiDou a septiembre del 2013 [Xingxing Li et. al, 2014].

- ## Wi-Fi positioning (WPS)

El posicionamiento WiFi utiliza puntos de acceso (APs) WiFi terrestres basados en Internet para determinar la ubicación [Zandbergen, 2009]. En los últimos años decenas de millones de APs que utilizan el estándar 802.11 han sido desplegados por individuos, propietarios, empresas, instituciones académicas, tiendas minoristas y edificios públicos. Todos estos APs repetidamente transmiten una señal anunciando su existencia a la zona circundante. Estas señales viajan típicamente varios cientos de metros en todas las direcciones. La densidad de APs en las áreas urbanas es tan alta que las señales a menudo se superponen, creando un sistema de referencia natural para determinar la localización. El software de posicionamiento WiFi identifica las señales WiFi existentes dentro del alcance de un dispositivo móvil habilitado

[24] http://mgex.igs.org/IGS_MGEX_Status_BDS.html (Último acceso: 10-dic-2016)

para WiFi y calcula la posición actual del dispositivo. La cobertura de posicionamiento WiFi es mejor en las zonas densamente pobladas. Los puntos de acceso WiFi se despliegan para uso privado y público para proporcionar cobertura inalámbrica de alta velocidad dentro de edificios y para áreas al aire libre seleccionadas. Como resultado, el posicionamiento WiFi en teoría tiene una excelente cobertura y rendimiento en interiores. Estos atributos lo distinguen del GPS, al cual le resulta dificultoso ofrecer información de posicionamiento en ambientes interiores. El posicionamiento WiFi no requiere que se establezca una conexión a la red WiFi: las señales WiFi sólo se registran en la forma de su dirección MAC única y la intensidad de la señal en una posición determinada. Esto permite que el posicionamiento WiFi utilice señales potencialmente muy débiles, así como señales cifradas, sin tener que establecer una conexión [Zandbergen, 2009]. Muchas veces lo complejo de este tipo de posicionamiento es contar con el registro de todas las señales en cada una de las posiciones, es decir, una grilla de señales. Muchas veces se logra solo dar un área dónde se puede encontrar la persona acorde al punto de acceso que recibió señal sin llegar a realizar una triangulación de todas las señales.

Como cada uno de los mecanismos mencionados previamente posee sus ventajas y desventajas, muchas veces se opta por un sistema hibrido de posicionamiento combinado en un mismo dispositivo, por ejemplo, el *A-GPS* y el *Wi-Fi positioning*. Esto es lo que se conoce como *Sistemas de posicionamiento híbridos* (XPS). La complementación de estas dos tecnologías permite lograr, en cualquier escenario, una mayor precisión.

La Figura 2.18 describe la relación entre las diferentes tecnologías que conforman el posicionamiento híbrido. Allí se puede ver que el posicionamiento GPS asistido por las torres de telefonía celular es lo que se denomina “A-GPS”; mientras que el Posicionamiento Híbrido (XPS) es el A-GPS asistido por Wi-Fi Positioning. Estas relaciones surgen como consecuencia de todos los conceptos mencionados anteriormente.

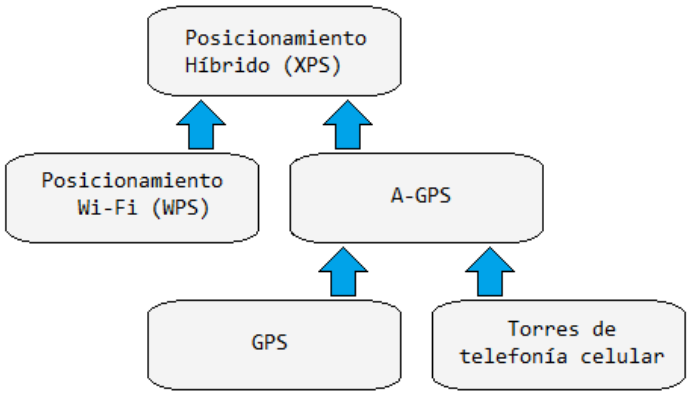


Figura 2.18: Relación entre mecanismos de posicionamiento.

La Figura 2.19 esquematiza la precisión de las tecnologías WPS y A-GPS operando en forma independiente y de su combinación (XPS), en entornos de diferentes densidades.

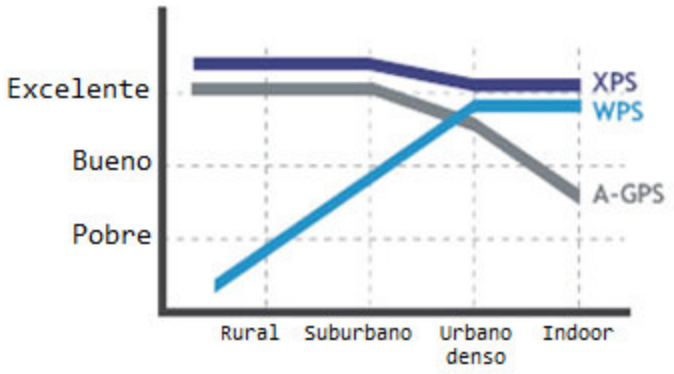


Figura 2.19: Precisión de respuesta de diferentes mecanismos de posicionamiento.[25]

Se puede observar en la Figura 2.19 los diferentes valores acorde al entorno. A continuación se brinda más nivel de detalle:

Rural: En este entorno, al tratarse de un espacio mayormente abierto el A-GPS tiene un desempeño excelente debido a la ausencia de obstáculos. El WPS tiene una precisión pobre debido a la ausencia de puntos de acceso Wi-Fi.

Suburbano: Los obstáculos presentes en este entorno no son suficientes para afectar la precisión del A-GPS, que se mantiene en excelente nivel. El WPS, por su parte, mejora su precisión debido al aumento de puntos de acceso Wi-Fi, llegando la misma a Buena.

Urbano denso: En este tipo de entornos, los obstáculos presentes (principalmente los denominados "cañones urbanos", conformados por aglomeraciones de edificios y rascacielos) reducen la precisión de A-GPS, mientras que el aumento de puntos de acceso Wi-Fi mejoran la precisión de WPS a un punto cercano a Excelente.

Indoor: En espacios interiores, la precisión del A-GPS se ve considerablemente perjudicada, mientras que la habitual existencia de puntos de acceso Wi-Fi le permiten al WPS mantenerse en un nivel próximo a Excelente.

2.3. Sistemas de seguimiento de mascotas

Los siguientes sistemas presentan características funcionales que buscan dar solución a la problemática de seguimiento de mascotas.

- **Cat@Log**

  En [Yonezawa, 2009] se describe el funcionamiento de *Cat@Log*, un sensor electrónico adherible para gatos (como puede verse en la Figura 2.16) para apoyar la interacción con

[25] Más información: http://www.avesnocturnas.es/2009/12/xps-el-sistema-hibrido-de-posicionamiento (Último acceso: 12-dic-2016)

humanos. Combina cámara, GPS, acelerómetro y Bluetooth. Este dispositivo puede reconocer las experiencias y actividades de gatos, información obtenida por los sensores y transmitida en tiempo real mediante el módulo inalámbrico Bluetooth.

Figura 2.20: Gato portando el dispositivo de sensado Cat@Log [Yonezawa, 2009].

Cat@Log utiliza Bluetooth y un software desarrollado para reconocer el comportamiento general de la mascota y publicarlo en Twitter, como se presenta en la Figura 2.21. Se puede observar que se hace una interpretación acorde a la actividad que está realizando el gato.

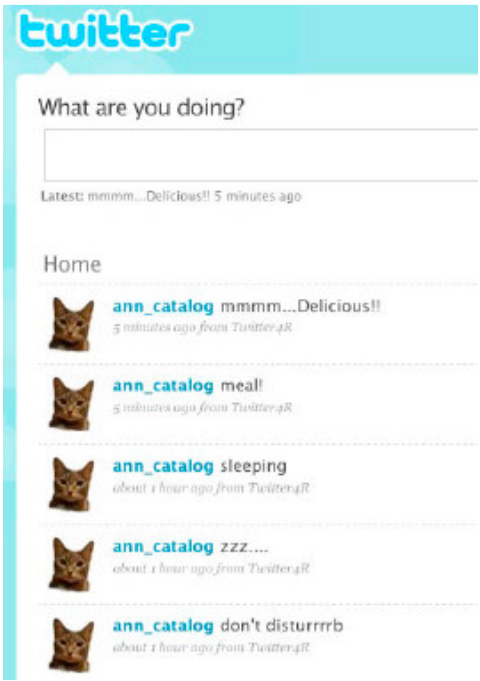


Figura 2.21: Publicaciones automáticas en Twitter realizadas por el dispositivo [Yonezawa, 2009].

- **Punetha-Mehta**

En [Punetha, 2014] se describe un sistema para trackear mascotas, niños, ancianos, personas con discapacidades o vehículos. El rango del sistema puede ser configurado en función del uso que se le dé. Si el portador del dispositivo atraviesa voluntaria o involuntariamente una determinada distancia, el sistema envía un SMS a un número predefinido revelando las coordenadas de su ubicación. La operación del sistema está basada en las redes GSM y satélites GPS. Este sistema no solo transmite latitud y longitud, sino también activa una alarma que indica el cruce de la periferia y le produce al portador una ligera descarga eléctrica para alertarlo (de esta manera lo describen los autores en [Punetha, 2014]). En la Figura 2.22 se puede apreciar los componentes involucrados en este sistema.

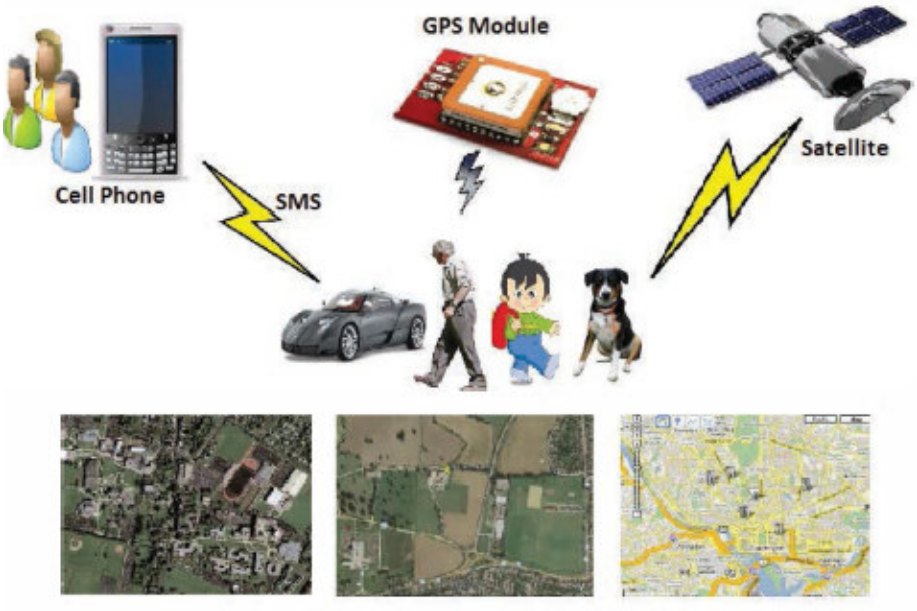


Figura 2.22: Componentes involucrados en el sistema presentado en [Punetha, 2014].

- **FindingRover**

Es una aplicación con reconocimiento facial de las mascotas (esta aplicación ya fue mencionada en el Capítulo 1). Utiliza algoritmos de reconocimiento facial para identificar la ficha de una mascota mediante la concordancia de su foto de perfil con una foto posterior. Para lograrlo, requiere que las fotos sean nítidas y frontales. Debido a la dificultad que representa fotografiar el rostro de una mascota, incluye un botón que reproduce un aullido agudo que llama la atención de la mascota hacia el dispositivo, permitiendo de esta forma fotografiar su rostro de frente. La Figura 2.23 muestra la pantalla de la aplicación en el momento de fotografiar a la mascota, señalando el botón que reproduce el aullido. Más información de dicha aplicación se puede apreciar en http://www.findingrover.com.

Figura 2.23: Captura de pantalla de FindingRover, presentando funcionalidad de fotografía de mascota, con botón de aullido.

- **ScanAPet**

  Es una empresa dedicada a la fabricación de chapas identificatorias (de acero inoxidable) que incluyen un código QR con un hipervínculo hacia una ficha creada por el usuario para la mascota (como ya se mencionó en el Capítulo 1). Además del código QR, la chapa incluye indicaciones breves para acceder a la ficha desde una PC, para el caso en que quien encuentra a la mascota no dispone de un *smartphone*. La ficha de la mascota puede incluir información sobre alimentación, alergias, dueño, veterinario, recompensa y mucha más. Ofrecen el mismo servicio para niños, equipaje, llaveros, tarjetas personales. El sistema no incluye ninguna funcionalidad de notificación. La Figura 2.24 muestra una pantalla con un ejemplo de ficha. Más información se puede apreciar en http://scanapet.com.

Figura 2.24: Ficha de mascota de ejemplo (extraída de http://scanapet.com).

- **PerrosQR**

  Empresa dedicada a la fabricación de chapas de identificación (realizadas en aleación de zinc, lo cual asegura una muy buena resistencia a la erosión) que incluyen un código QR que codifica un hipervínculo hacia una ficha con información de la mascota a elección del cliente, como por ejemplo información de contacto (como ya se mencionó en el Capítulo 1). El sistema no incluye ninguna funcionalidad de notificación. En la Figura 2.25 se presenta un ejemplo de ficha. Más información se detalla en http://www.perrosqr.com.

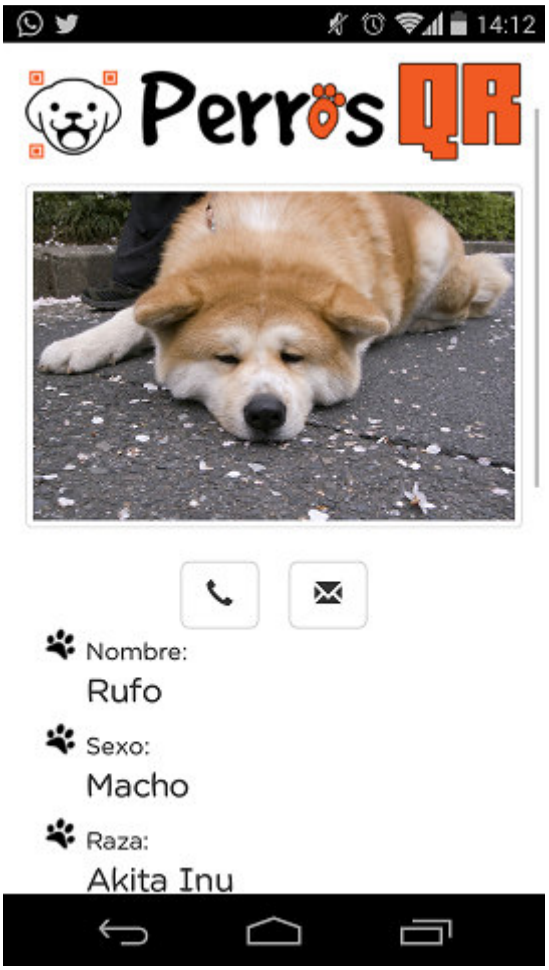


Figura 2.25: Ficha de mascota de ejemplo (extraída de http://www.perrosqr.com)

- **Registro e identificación de perros callejeros de Berazategui**[26]

  La Clínica Veterinaria de Berazategui implementó un sistema de registro e identificación de los perros comunitarios de la ciudad. El mismo consiste en la colocación de un dispositivo-botón (un distintivo circular que se coloca en la oreja del animal, como muestra la Figura 2.26), el cual posee un número de registro a través del cual se puede acceder a toda su información (fechas de vacunación, castración y desparasitación).

  Según indica el director de la Clínica Veterinaria Municipal, Luis Martínez, se coloca un dispositivo de plástico muy liviano en una de las orejas del perro, el mismo que se usa en otras especies como vacas u ovejas, el cual no genera molestias ni complicaciones para los perros. El mismo está numerado y la Clínica Veterinaria Municipal tiene un registro para, a simple vista, identificar su situación: cuándo fue castrado, vacunado o desparasitado. Gracias a eso, por

[26] http://www.berazategui.gob.ar/noticias/110-general/553-los-perros-de-la-calle-ya-tienen-su-propio-registro-en-berazategui

cualquier inconveniente que surja con ese animal, los vecinos se pueden comunicar con la Clínica Veterinaria Municipal, comunicar el número que tiene en la oreja ese perro para que ellos puedan recurrir a la información del mismo.

El principal objetivo de la iniciativa es que los perros que se encuentran en las calles de la ciudad tengan el cuidado, el control y el seguimiento necesario para que se desarrollen sanos y saludables.

Figura 2.26: Perro portando en su oreja el botón identificatorio [19].

A continuación se presenta un resumen de las principales características de los sistemas o aplicaciones descriptas anteriormente. Estas características se pueden apreciar en la Tabla 2.7.

Tabla 2.7: Extracto de las características principales de los sistemas anteriores.

Sistema	Información de posicionamiento	Alertas automáticas	Identificación de mascota	Servicios adicionales
Cat@Log	Continua	Twitter	No	Bitácora de la mascota
Punetha-Mehta	Al salir de área	SMS	No	No
FindingRover	No	No	Reconocimiento facial	No
ScanAPet	No	No	Código QR	No
PerrosQR	No	No	Código QR	No
Registro Berazategui	No	No	Número	No

Se puede apreciar en la Tabla 2.7 que no todos los sistemas poseen información de posicionamiento, y aquellos que sí, no poseen identificación de mascotas. En la tesis propuesta se hará hincapié en brindar soporte a ambos aspectos.

Capítulo 3. Modelo propuesto

En este capítulo se detallará la problemática para la cual se busca una solución de modelado que la resuelva. Complementariamente a la descripción de la problemática se incorporará el detalle de los casos de uso considerados. Luego, se detallará la solución de modelado propuesta en esta tesis.

3.1. Caracterización de la problemática a resolver

Como se mencionó en la Sección 1.1 existen diversos sistemas y empresas que ofrecen servicios para facilitar la recuperación de una mascota (algunas fueron detalladas en la Sección 2.3). Sin embargo, de las soluciones analizadas, se encontró que ninguna provee una solución de modelado para la problemática del extravío de mascotas. Por otra parte, el alcance de estos sistemas es muy acotado en términos funcionales, no siendo extensibles para funcionalidades complementarias con miras al bienestar de las mascotas.

La problemática que se busca resolver en esta tesis es el diseño de un modelo que permita representar a las mascotas con sus características, su posicionamiento, el estado en que se encuentra respecto a su hogar, y el historial de eventos registrados sobre dicha mascota. También se buscará representar la recompensa ofrecida por la reunión de una mascota encontrada con su dueño. El modelo deberá contemplar mecanismos de notificación automáticos al dueño de la mascota ante el hallazgo de la misma por parte de cualquier persona, como así también mecanismos de generación de códigos bidimensionales que permitan identificar unívocamente a la mascota. El correcto modelado de las características mencionadas haría posible desarrollar un sistema que permita notificar en tiempo real a un usuario el extravío o aparición de su mascota, al ser la misma identificada por cualquier persona que encuentre su identificación.

Adicionalmente, se buscará demostrar la extensibilidad del modelo al complementarlo con la representación de necesidades asociadas a las mascotas, y recursos asociados al usuario, que le permitan al sistema ofrecer la funcionalidad de conectar mascotas con voluntarios (personas que ofrecen, por ejemplo, medicamento para que otros lo puedan usar).

El modelo propuesto permitiría la construcción de un sistema que, una vez implementado, facilitaría la solución de las siguientes problemáticas:

- *Recuperación de mascotas extraviadas*.
- *Reubicación de mascotas abandonadas*: así como se puede determinar la posición de un animal extraviado, se puede publicar la ubicación de un animal que necesita una nueva familia.
- *Asistencia a animales con necesidades*: un animal puede requerir recursos que su familia difícilmente puede proveerle, como medicación, alimento, tratamientos, materiales para

refugios, prótesis, entre otros. Asociar las necesidades a la mascota y publicar su ubicación permitiría que otros usuarios se acercaran para colaborar.

- *Ofrecimiento de recursos por parte de usuarios*: muchas personas tienen intención de colaborar con animales mediante sus recursos (económicos, profesionales, materiales, de traslado), pero no disponen de tiempo o movilidad para presentarse en un refugio, o bien el aporte que pueden hacer es tan humilde que no amerita movilizarse. Así como un usuario puede registrar en el sistema su ubicación geográfica, también puede publicar recursos de los que dispone.

De la problemática descripta se desprenden diversos conceptos y relaciones que necesitan ser contemplados en el modelado del sistema:

Estamos considerando que cada *Mascota* tiene un *Dueño*, que intentaría reencontrarse con ella a través del *Sistema* en caso de estar *Extraviada*. Esta mascota puede ser *Encontrada* por otra persona, gracias a la cual puede ser *Recuperada*. Estos cambios de *Estado* se producen ante *Eventos*, que se notifican al sistema en un momento (Cuándo), en un lugar (Dónde) por una persona (Quién). Y las personas pueden requerir que el sistema funcione como un *Notificador* de estos eventos en relación a su mascota. Mientras la mascota está extraviada, es posible que su dueño quiera incentivar la colaboración de la gente mediante una *Recompensa*.

Por otra parte, hemos mencionado que una mascota (extraviada o no) puede padecer *Necesidades* de diferentes *Tipos* (medicación, alimentos, materiales para refugio, abrigo, tratamiento, prótesis, traslado o incluso familia), mientras que las personas pueden disponer de *Recursos* que vengan a satisfacer las necesidades identificadas.

Estos conceptos identificados formarán parte del modelo propuesto como se presenta más adelante en este capítulo.

En la Figura 3.1 se puede apreciar los diferentes casos de uso identificados los cuales servirán como base para plantear el modelo propuesto en la siguiente sección. Se puede apreciar que hay dos actores bien diferenciados, los usuarios registrados y aquellos no registrados. El estar registrados en el sistema permite tener otros servicios relacionados con sus mascotas como se puede apreciar en dicha figura.

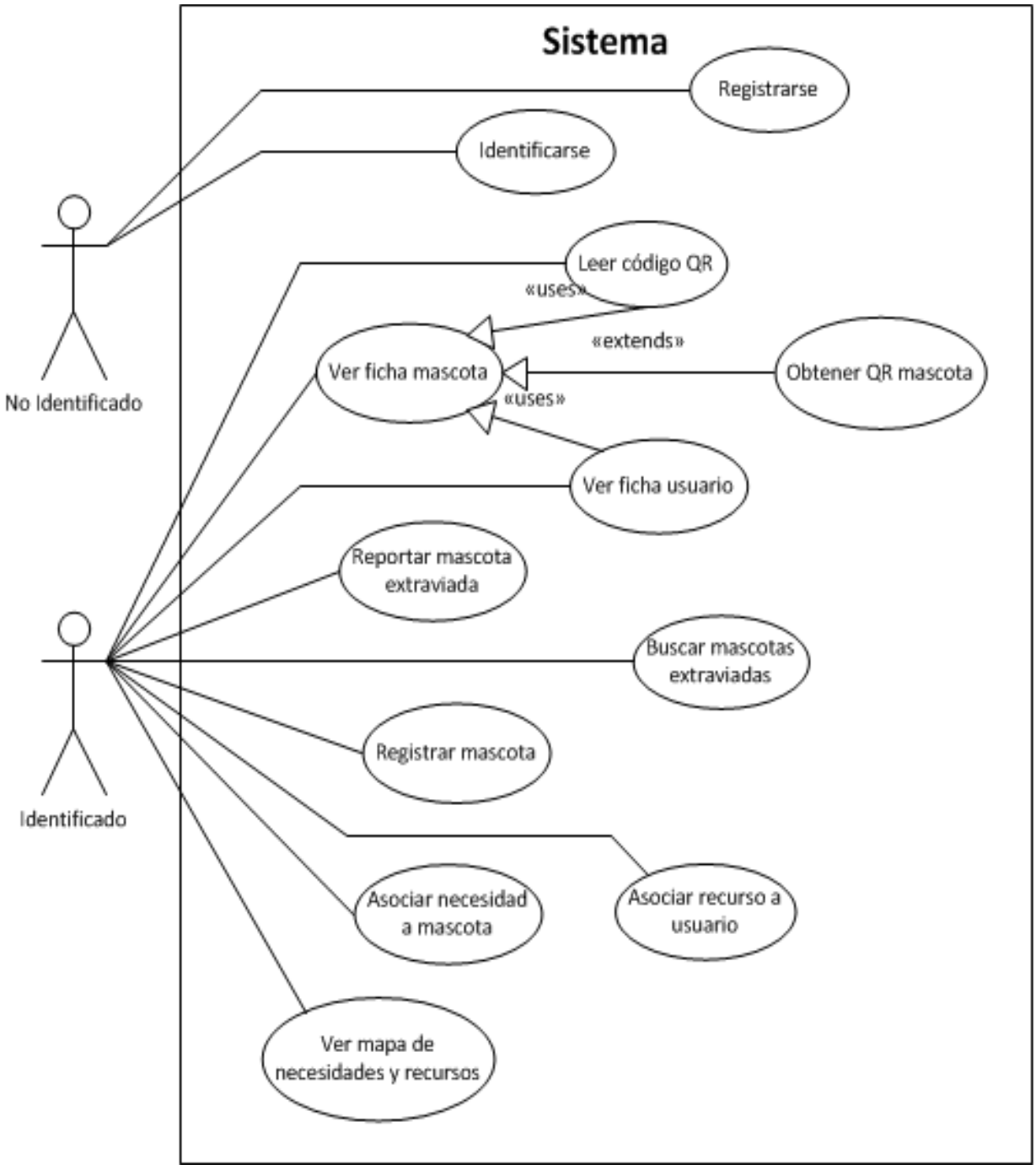


Figura 3.1: Diagrama de casos de uso.

3.2. Descripción del Modelo Propuesto.

A partir de lo detallado en la sección previa, en esta se presenta el modelo propuesto de manera incremental. El modelo propuesto cuenta con una fachada para brindar la funcionalidad, cumpliendo con lo descripto en los patrones *Facade*[27] [Gamma et. al, 2003] y *Singleton*[28] [Gamma et. al, 2003]. Esta fachada lleva el nombre de *Sistema* y se puede apreciar en la Figura 3.2.

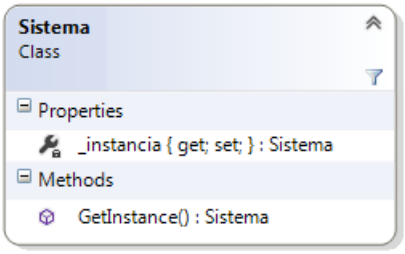


Figura 3.2: Clase Sistema, como punto de entrada a la funcionalidad del modelo.

La clase *Sistema* conoce cuales todos los tipos de mascota del sistema, como por ejemplo "*Gato*", "*Perro*", "*Hurón*", "*Lagarto*", etcétera. Dichos tipos son representados por la clase *TipoAnimal*. Esto se puede apreciar en la Figura 3.3.

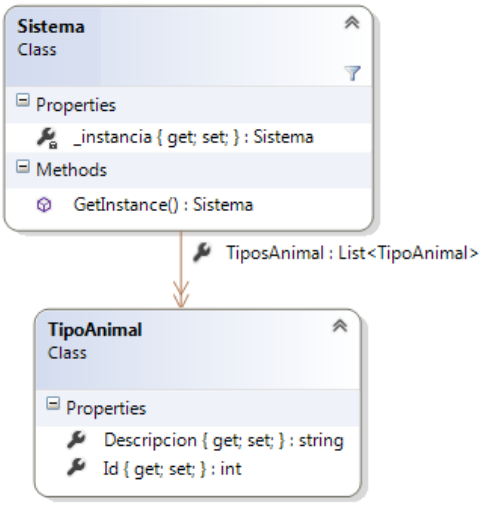


Figura 3.3: Se incorpora la clase *TipoAnimal*.

Por otra parte, la clase *Sistema* conoce a todos los usuarios, donde cada *Usuario* tiene información de: su nombre, su apellido, teléfono, dirección de email, su última posición conocida y una forma de identificarlo (id) en el sistema. El Sistema posee un método para obtener un usuario dado un id de usuario, esto se puede apreciar en la Figura 3.4. Se puede observar que la posición del usuario

[27] *Facade*: Proporciona una interfaz unificada para un conjunto de interfaces de un subsistema. Define una interfaz de alto nivel que hace que el subsistema sea más fácil de usar.
[28] *Singleton*: Garantiza que una clase solo tenga una instancia, y proporciona un punto de acceso global a ella.

se representa con una clase denominada *LatLon*, que representa una latitud y longitud. Por simplificación se eligió esta representación, pero el modelo podría extenderse para considerar otras clases de posiciones.

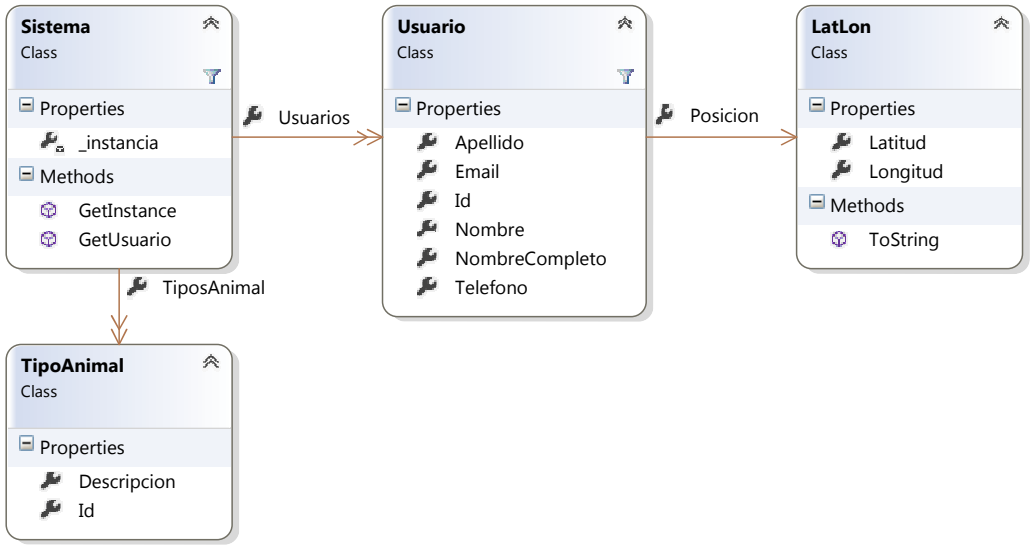


Figura 3.4: Se incorpora la clase *Usuario*, y los métodos de *Sistema* para obtener *Usuario* por id.

El *Sistema* también conoce a todas las mascotas registradas, donde cada *Mascota* tiene un nombre, una fecha de nacimiento, un tipo de animal, un dueño y un identificador (id) dentro del *Sistema*. Cada *Mascota* conoce cuál es su edad, y puede responder si un usuario es su dueño o no. Lo antes descripto se puede apreciar en la Figura 3.5.

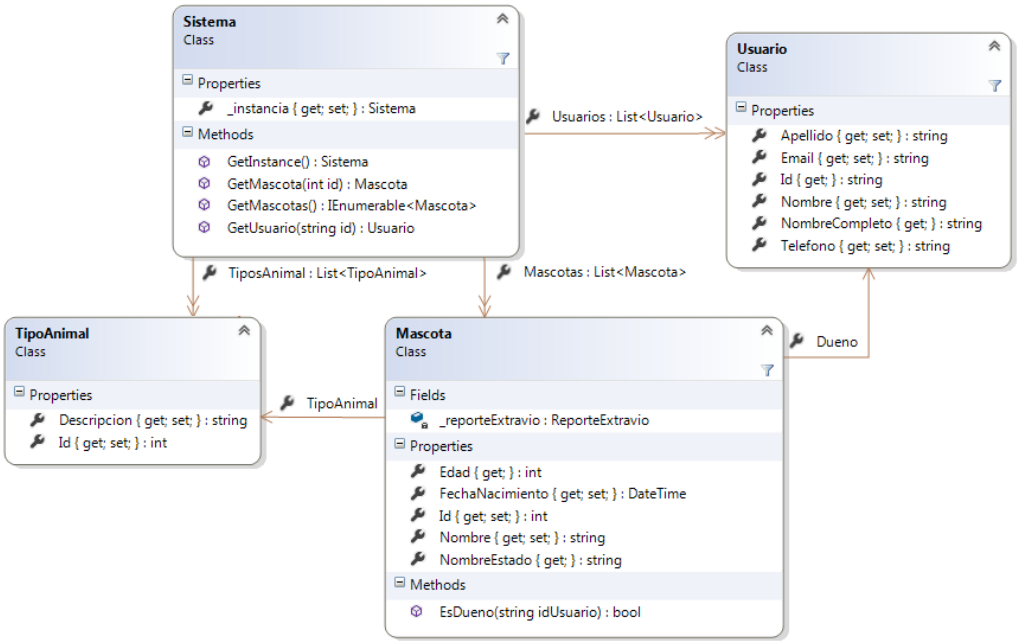


Figura 3.5: Se incorpora la clase *Mascota*, y en *Sistema* los métodos para obtenerlas todas, y obtener *Mascota* por Id.

Cada *Usuario* tiene asociado un mecanismo de notificación preferido. Una notificación es un aviso o alerta que el *Sistema* necesita transmitirle a un usuario en particular. El modelo define una jerarquía de notificadores representada por una superclase abstracta *Notificador* y (como ejemplo) una subclase *NotificadorSMS* que implementa el método de notificación de un texto a un usuario. Esta estructura cumple con el patrón *Strategy*[29] [Gamma et. al, 2003]. Esta jerarquía se puede apreciar en la Figura 3.6, donde en particular por ahora se especificó una sola subclase pero podría haber otras estrategias de notificación. Esto es un punto de extensión del modelo propuesto.

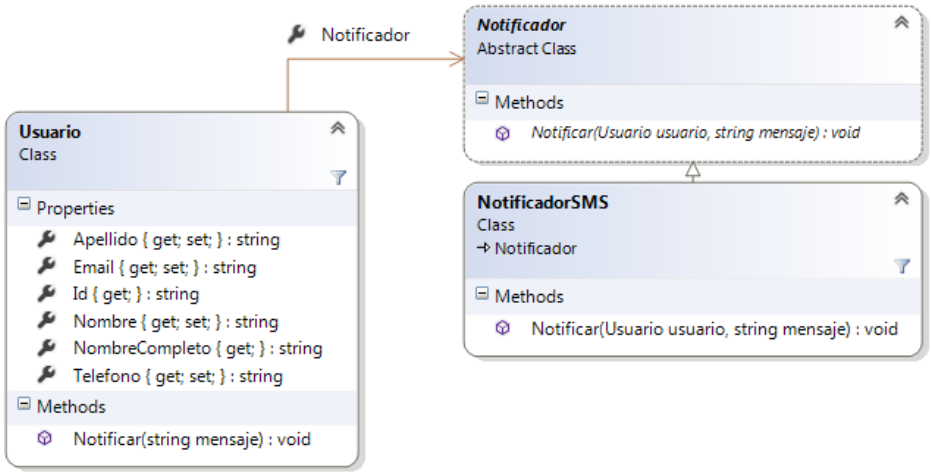


Figura 3.6: Se incorpora la jerarquía Notificador, con la implementación concreta *NotificadorSMS*.

Por otra parte, el modelo contempla el registro temporal y posicionado de diferentes eventos mediante la clase *RegistroPosicionUsuario* (ver Figura 3.7), donde cada registro está conformado por Cuándo (fecha y hora), Quién (usuario) y Dónde (posición). Como se mencionó anteriormente, en este modelo la posición está representada por la clase *LatLon*, que representa una latitud y longitud.

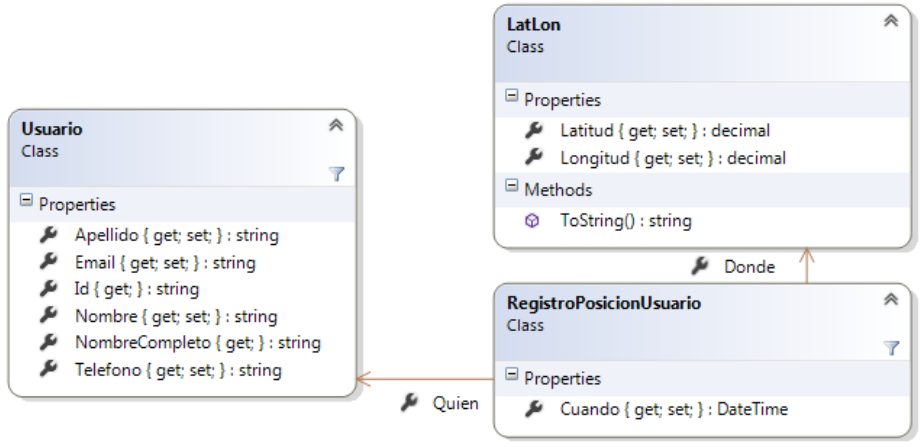


Figura 3.7: Se incorpora la clase *RegistroPosicionUsuario*.

[29] *Strategy*: Define una familia de algoritmos, encapsula cada uno de ellos y los hace intercambiables. Permite que un algoritmo varíe independientemente de los clientes que lo usan.

El modelo contempla la representación de estados de extravío de las mascotas, como *Encontrado*, *Extraviado* y *Recuperado*. Dichos estados están representados por una jerarquía conformada por la superclase abstracta *EstadoExtravío* (ver Figura 3.8), donde cada estado tiene asociado uno de los ya descriptos registros de posición, y su estado previo (un estado de la jerarquía), conformando una posible estructura recursiva. Esta estructura recursiva se utilizó para lograr el patrón *Chain of Responsibility*[30] [Gamma et. al, 2003], que le permite al modelo obtener la fecha en que se inició la cadena de estados, como así también determinar (con el método recursivo *ConoceA*) si un determinado usuario encontró alguna vez a la mascota. El método *EstadoConoceA* (utilizado por *ConoceA*, como se verá luego en diagramas de secuencia) no es recursivo, sino que responde solo por la instancia concreta de *EstadoExtravio*.

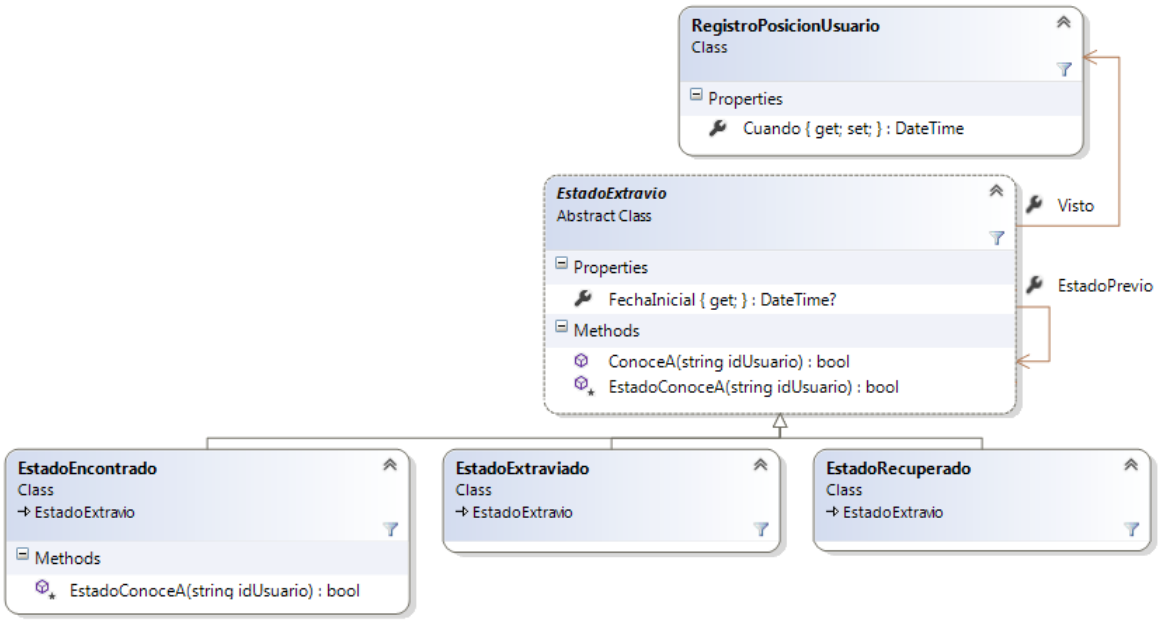


Figura 3.8: Se incorpora la jerarquía *EstadoExtravio*, con referencia a *RegistroPosicionUsuario*.

Los mencionados estados de la Figura 3.8 se encuentran asociados a la mascota a través de los reportes de extravío. Los mismos están modelados por la clase *ReporteExtravio*. Una mascota tendrá asociado un *ReporteExtravío* desde el momento en que mediante una acción de usuario el sistema determine que la mascota está extraviada. Cada reporte puede registrar (opcionalmente) la recompensa determinada por el usuario por la recuperación de su mascota extraviada. El reporte también cuenta con un estado actual, es decir, una referencia al eslabón más reciente de la cadena de estados descripta previamente en la Figura 3.8. Como los Estados cambian el comportamiento del reporte ante determinado evento, la jerarquía *EstadoExtravio* cumple con el patrón *State*[31] [Gamma et. al, 2003]. El *ReporteExtravío* hace uso de la cadena de estados para

[30] *Chain of Responsibility*: Evita acoplar el emisor de una petición a su receptor, dando a más de un objeto la posibilidad de responder a la petición. Encadena los objetos receptores y pasa la petición a través de la cadena hasta que es procesada por algún objeto.

[31] *State*: Permite que un objeto modifique su comportamiento cada vez que cambie su estado interno. Parecerá que cambia la clase del objeto.

responder si un usuario estuvo involucrado en alguno de los cambios de estado. Esta nueva clase incorporada se puede apreciar en la Figura 3.9.

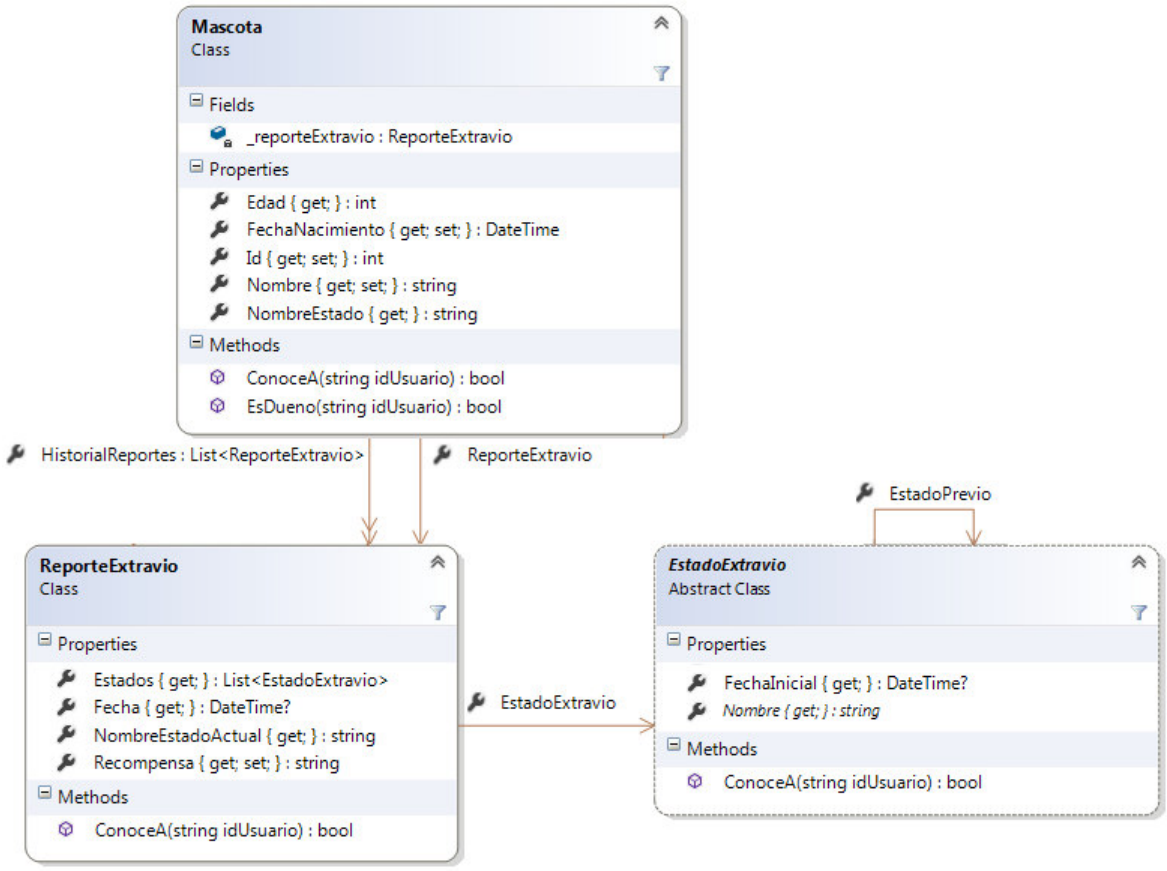


Figura 3.9: Se incorpora la clase *ReporteExtravio*.

Los cambios de estado de los reportes se producen ante eventos, que están representados por la jerarquía *Evento*, y los mismos interactúan con la jerarquía *EstadoExtravio* para lograr los cambios de estado y disparar las notificaciones a usuarios. Esta interacción entre jerarquías cumple con lo descripto en el patrón *Visitor*[32] [Gamma et. al, 2003]. Entre los eventos se encuentran el anuncio de extravío de mascota por parte del dueño; la lectura del código QR por parte de un desconocido o la lectura de un código QR por parte del dueño, como se puede apreciar en la Figura 3.10.

Los métodos que pueden encontrarse en dicha jerarquía son sobrecargas del método Visitar, donde cada una recibe un tipo de *EstadoExtravio* específico. Como se verá más adelante, el método "*Aceptar*" de cada *EstadoExtravio* concreto invocará al método "*Visitar*" del Evento capturado enviándose a sí mismo como parámetro, lo cual determina qué sobrecarga de "*Visitar*" se ejecuta, y, por lo tanto, qué procesamiento realizar para la interacción entre esta combinación de Estado y Evento.

[32] *Visitor*: Representa una operación sobre los elementos de una estructura de objetos. Permite definir una nueva operación sin cambiar las clases de los elementos sobre los que opera.

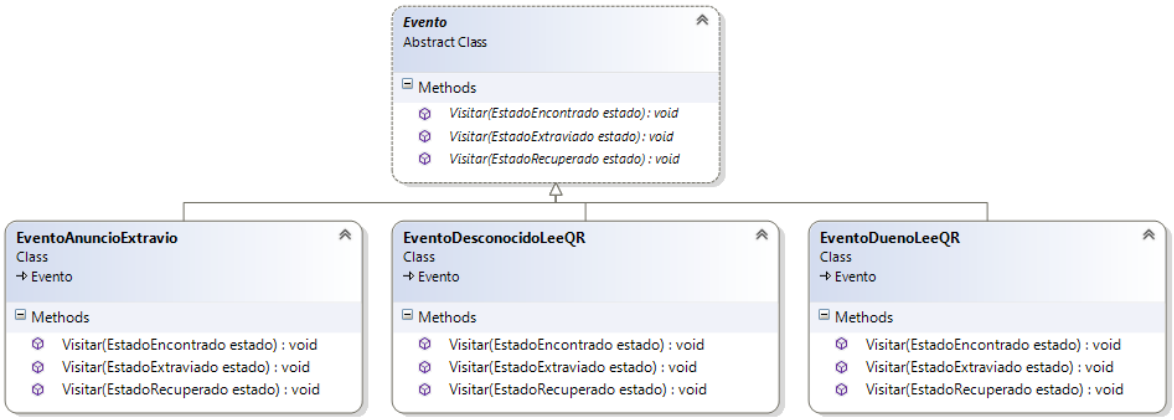


Figura 3.10: Se incorpora la jerarquía *Evento*.

Cada clase de la jerarquía *Evento* cuenta con la mascota por la cual se produjo dicho evento, como así también al registro de posición en el cual se produjo el mismo. Esto se puede apreciar en la Figura 3.11.

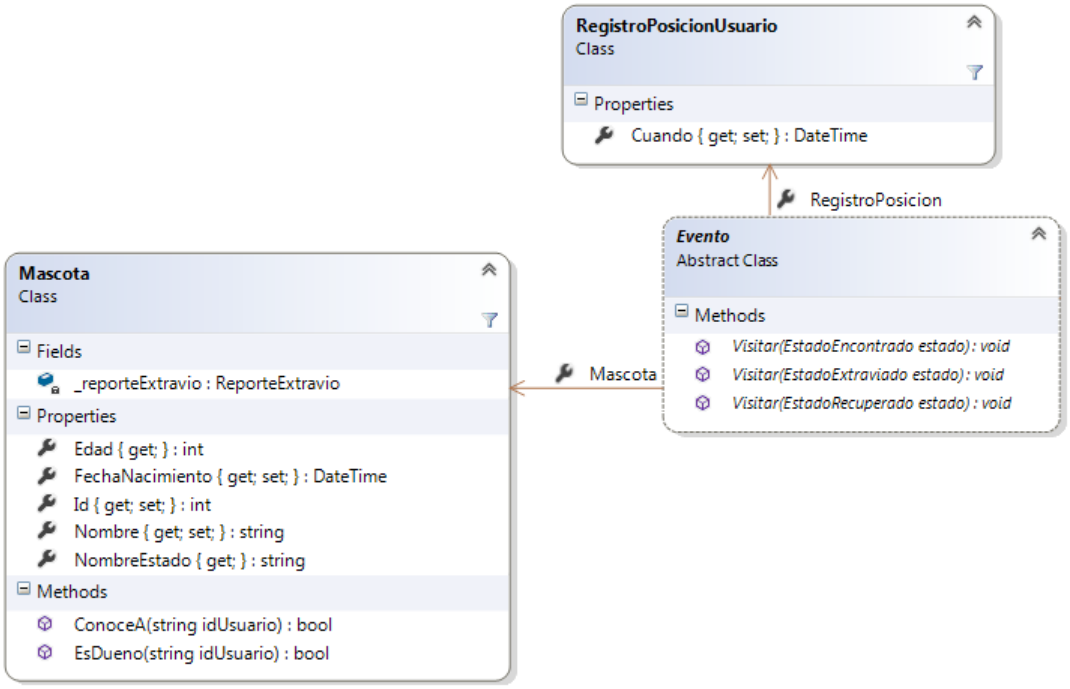


Figura 3.11: Se incorporan las relaciones desde *Evento* a *Mascota* y *RegistroPosicionUsuario*.

Como se mencionó anteriormente, un reporte de extravío se instancia cuando el sistema determina mediante algún evento que la mascota está extraviada. Por ejemplo, cuando el dueño de la mascota reporta que la misma está extraviada (*EventoAnuncioExtravio*) o cuando un desconocido lee el código QR de la mascota (*EventoDesconocidoLeeQR*). Si bien ambos casos producen la instanciación de un *ReporteExtravio*, el estado inicial de los mismos será diferente (*EstadoExtraviado*, *EstadoEncontrado*, respectivamente).

Conceptualmente la lógica para determinar un estado inicial de reporte no es muy diferente de la forma en que se determinan posteriores cambios de estado, mediante la interacción de *Eventos* y *Estados*. Pero al instanciarse un reporte con *Estado* nulo, no es posible hacer interactuar el *Evento* con el nulo de la misma forma que se lo hace interactuar con *Estados*. Para resolver este problema se utilizó el patrón *Null Object*[33], agregando a la jerarquía *EstadoExtravío* un *EstadoNull*, con el cual es inicializado todo *ReporteExtravio*. Esto se puede apreciar en la Figura 3.12.

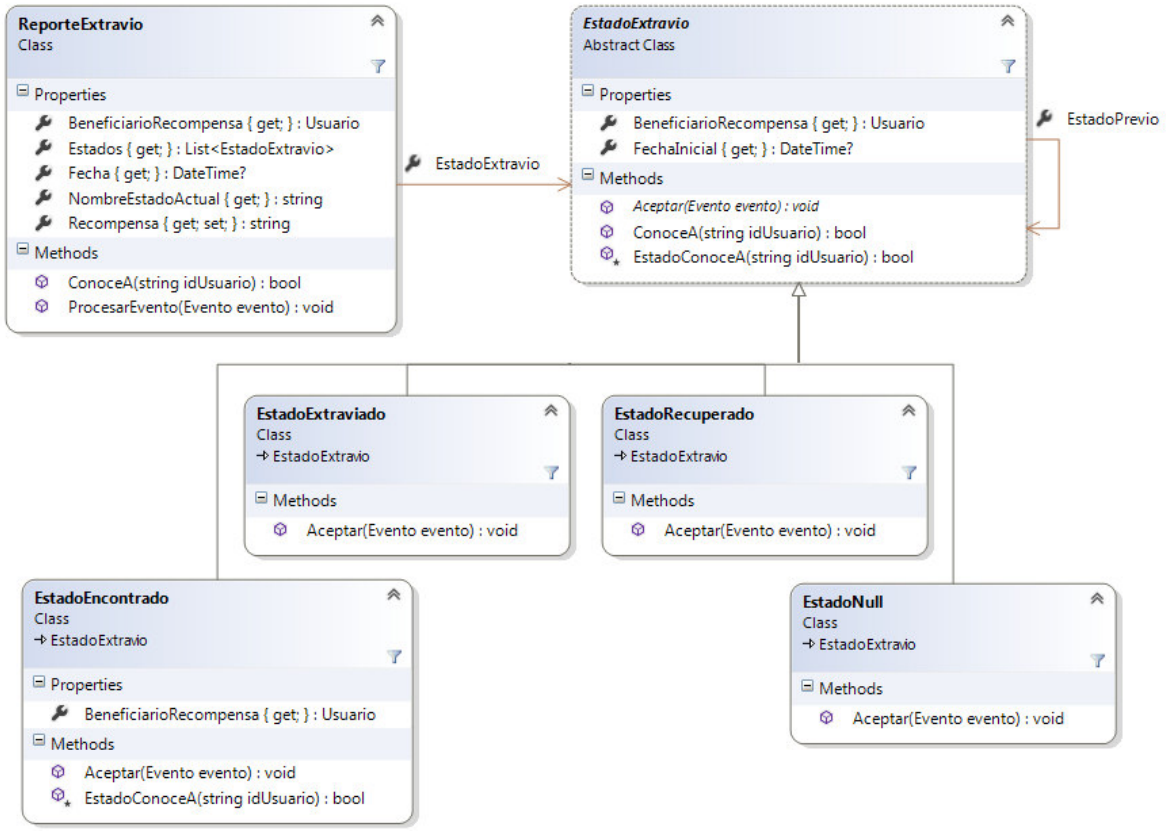


Figura 3.12: Se incorpora la subclase *EstadoNull* en la jerarquía *EstadoExtravio*..

En este punto es importante mencionar que el mecanismo de recompensas por hallazgo de mascotas hace uso de la cadena de estados para determinar, al momento de reunión del dueño con su mascota, quién es el beneficiario de la recompensa. Será el usuario asociado al evento *EstadoEncontrado* más reciente de la cadena.

Al haberse incorporado un nuevo estado en la jerarquía *EstadoExtravio*, es necesario incorporar en la jerarquía de *Eventos* la lógica de tratamiento del mismo como se puede apreciar en la Figura 3.13.

[33] Null Object: Encapsular la ausencia de un objeto mediante la provisión de una alternativa sustituible que ofresca un valor por defecto con comportamiento "No hacer nada".

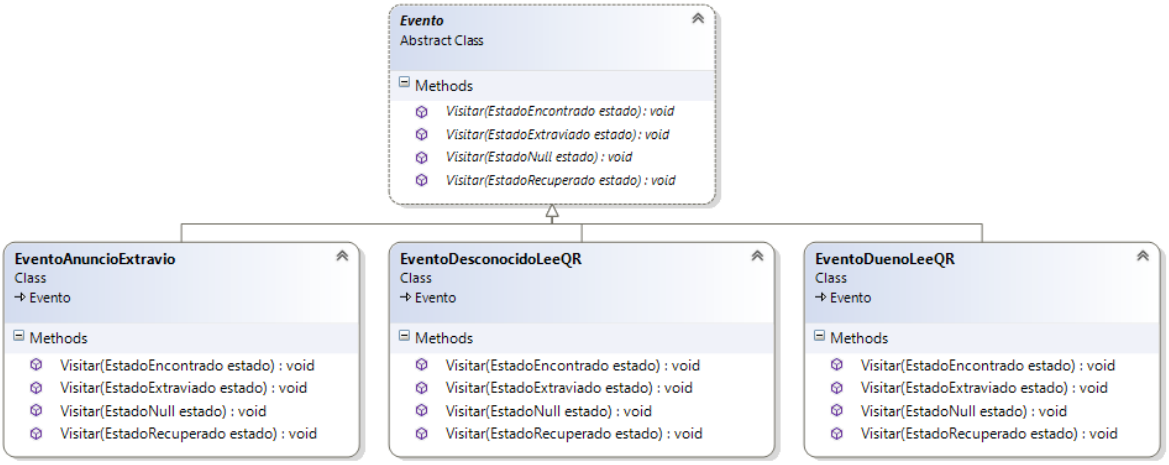


Figura 3.13: Se incorpora a cada Evento la sobrecarga de *Visitar* que contempla el *EstadoNull*.

La descripción del modelo ya realizada hasta ahora permite incorporar a la clase *Sistema* los métodos más importantes, encargados de procesar la lectura de un código QR de una mascota y procesar la determinación del usuario de que su mascota se extravió. Estos se pueden observar en la Figura 3.14.

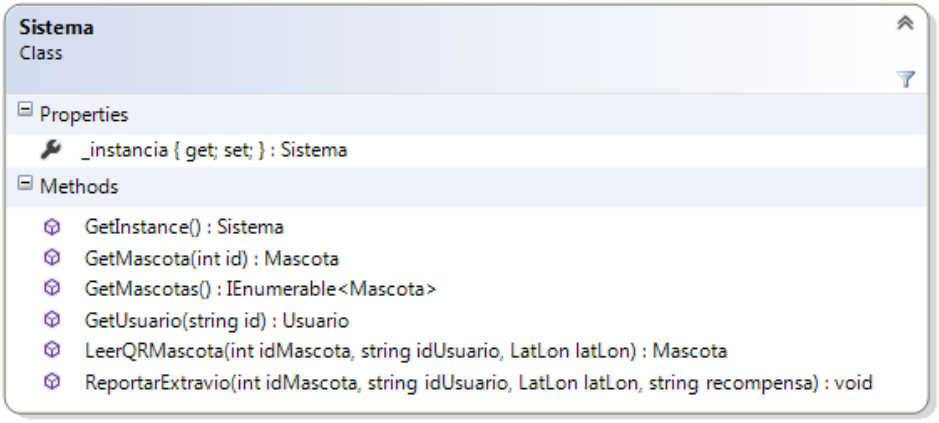


Figura 3.14: Se incorpora en la clase *Sistema* los métodos *LeerQRMascota* y *ReportarExtravio*.

Por último, el modelo propuesto contempla la posibilidad de que el mecanismo de generación de código bidimensional esté desacoplado en una jerarquía de generadores. Está representado por la jerarquía *GeneradorCodigo2D*, y su utilización le permite a la clase *Sistema* brindar el método para generar el código bidimensional. El desacoplamiento de este comportamiento configurable cumple con lo definido en el patrón *Strategy* [Gamma et. al, 2003]. Lo antes descripto se presenta en la Figura 3.15

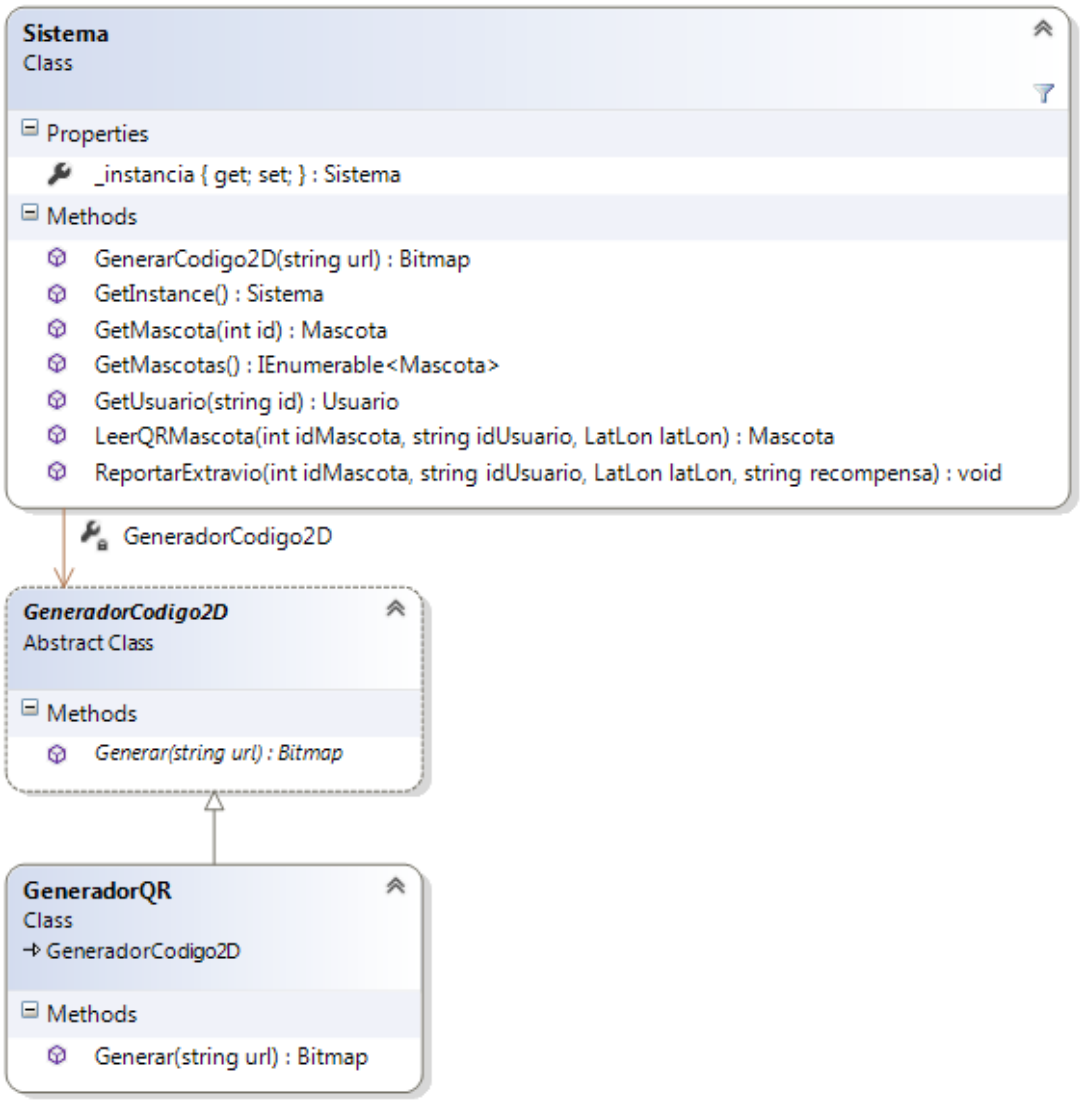


Figura 3.15: Se incorpora la jerarquía *GeneradorCodigo2D*.

Acorde a todos los conceptos descriptos, en la Figura 3.16 se presenta el modelo propuesto completo.

Figura 3.16: Modelo completo.

Para una mejor comprensión de cómo se dan la transiciones entre los estados mencionados en el modelo propuesto, en la Figura 3.17 se presenta un diagrama de transición de estados. En este diagrama se presentan los flujos entre los cuatro estados de extravío (*EstadoNull, EstadoExtraviado, EstadoEncontrado, EstadoRecuperado*) ante el procesamiento de los tres eventos del modelo (*EventoAnuncioExtravio, EventoDueñoLeeQR, EventoDesconocidoLeeQR*).

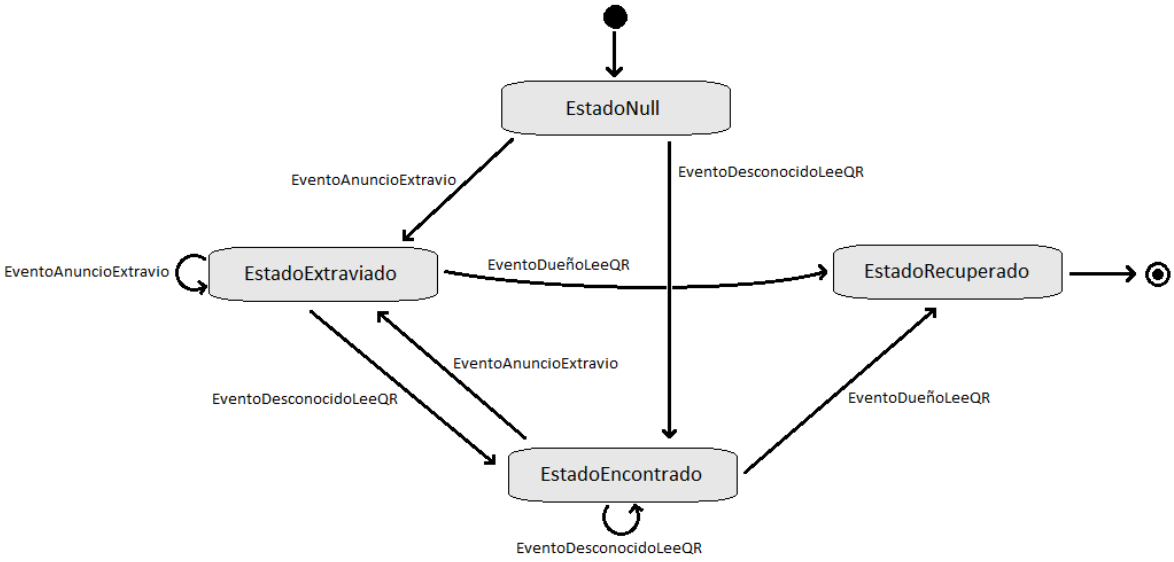


Figura 3.17: Diagrama de estados.

En la Figura 3.17 se puede apreciar que del estado *EstadoNull,* según se haya dado el evento *EventoAnuncioExtravio* (que lo dispara el propio dueño) *o EventoDesconocidoLeeQR,* esto puede generar que se pase a los estados *EstadoExtraviado o EstadoEncontrado* respectivamente. Tanto del estado *EstadoExtraviado* como *EstadoEncontrado* cuando se da el evento de *EventoDueñoLeeQR* se pasa al estado *EstadoRecuperado.* Mientras que siempre que se da un evento *EventoDesconocidoLeeQR* se pasa al estado *EstadoEncontrado.* El bucle en *EstadoEncontrado* se debe a que podría pasar que más de una persona encuentre la mascota y la misma no haya podido ser retenida, y en este caso queda registro de por dónde fue vista.

Tomando como base el diagrama de estados presentado en la Figura 3.17, se presentan diferentes diagramas de secuencias que describen las operaciones más importantes sobre el modelo.

- *Reportar Extravío*

  Se puede apreciar en la Figura 3.18 el diagrama de secuencia que se desencadena cuando el dueño de una mascota reporta su extravío. Se registra la posición del mismo y se genera un evento de reporte de extravío.

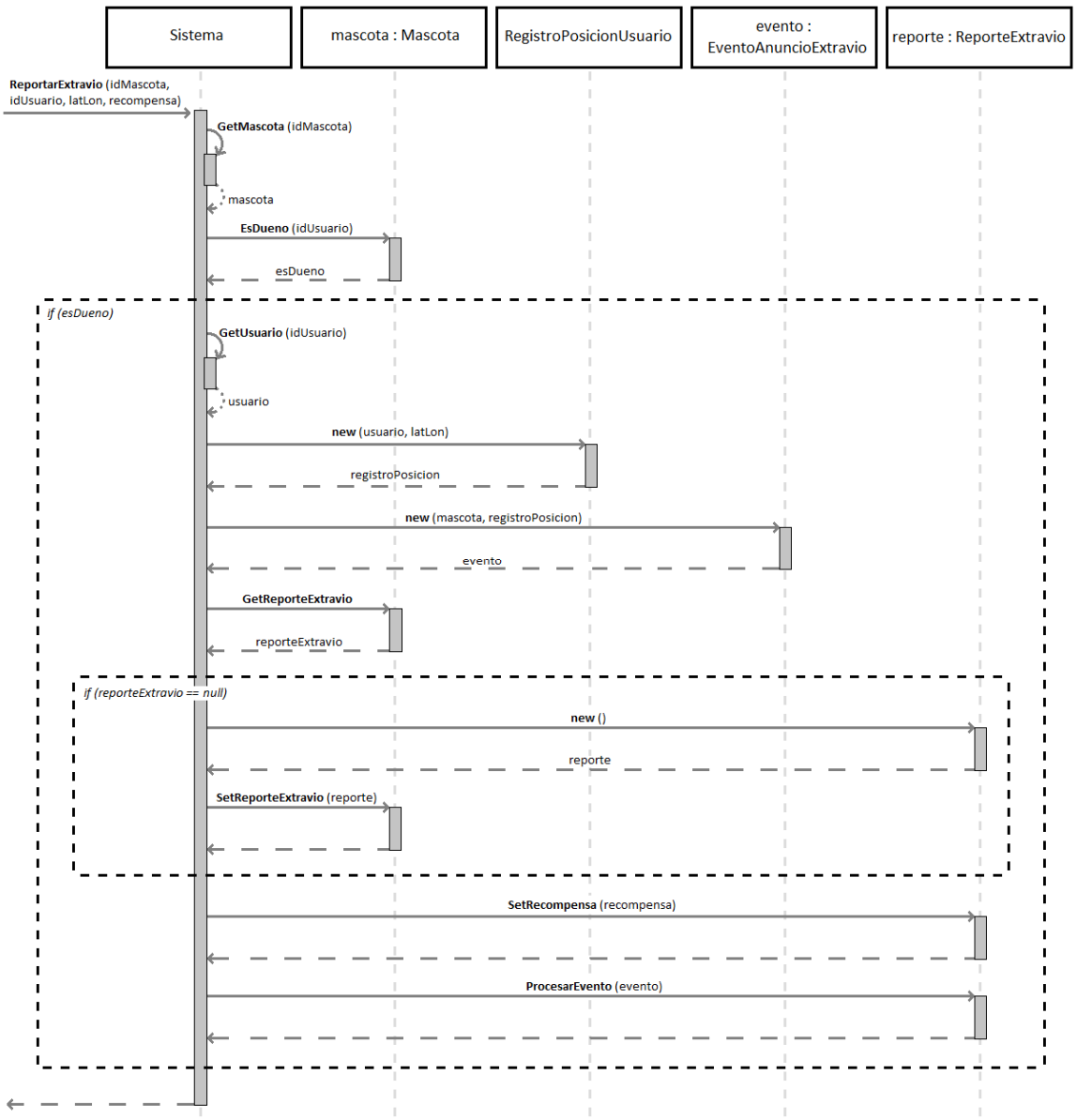


Figura 3.18: Diagrama de secuencia del método `ReportarExtravio()` de Sistema

- *Leer código QR de mascota*

  En la Figura 3.19 se describe el diagrama de secuencia desencadenado al leer (un usuario cualquiera) el código QR de una mascota. Como la acción de leer un código QR puede ser realizada por cualquier usuario, las acciones que se desencadenan dependerán de si la lectura es del dueño o no. El método `ObtenerEventoLecturaQR()` es el que permite luego determinar, por ejemplo, si ha sido el dueño el que leyó el código QR o no. En ambos casos, como se puede visualizar en esta figura, se genera (de no existir previamente) un *ReporteExtravio* para que procese el evento.

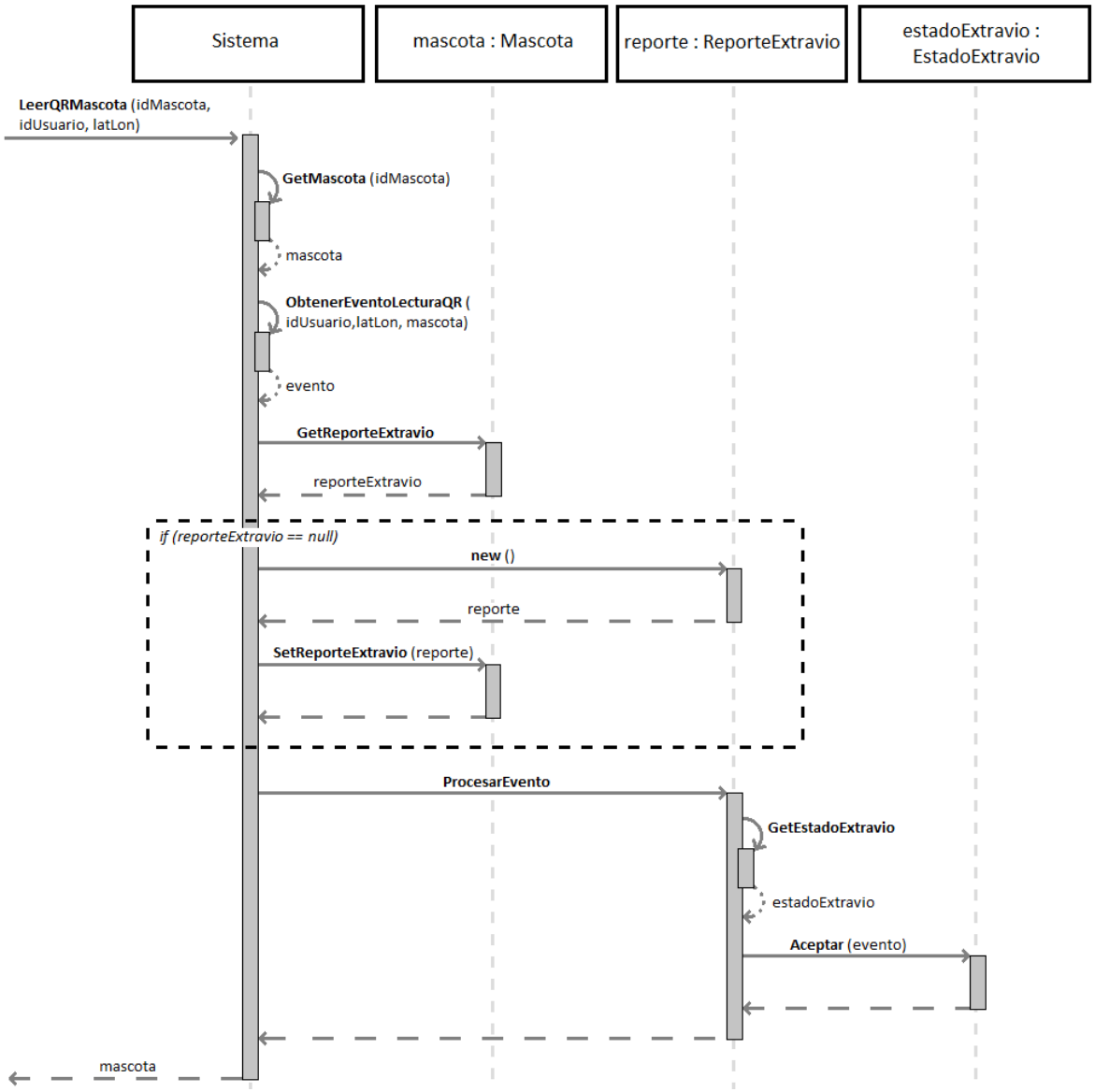


Figura 3.19: Diagrama de secuencia del método `LeerQRMascota()` de Sistema.

- *Leer código QR de mascota ➔ Subdiagrama: Obtención de evento de lectura*

  Como se indicó anteriormente, el diagrama de secuencia presentado en la Figura 3.19 corresponde a la lectura de un código QR por parte de cualquier usuario. Sin embargo, el Evento disparado por esta acción depende de si el usuario es el dueño de la mascota o no. Esto es determinado (como fue mencionando anteriormente) por la invocación al método `ObtenerEventoLecturaQR()`. En la Figura 3.20 se muestra la secuencia de mensajes de este método tanto cuando el usuario es dueño de la mascota como cuando no. En cada caso, se generan eventos diferentes, y acorde a eso son las secuencias de mensajes que se desencadenan.

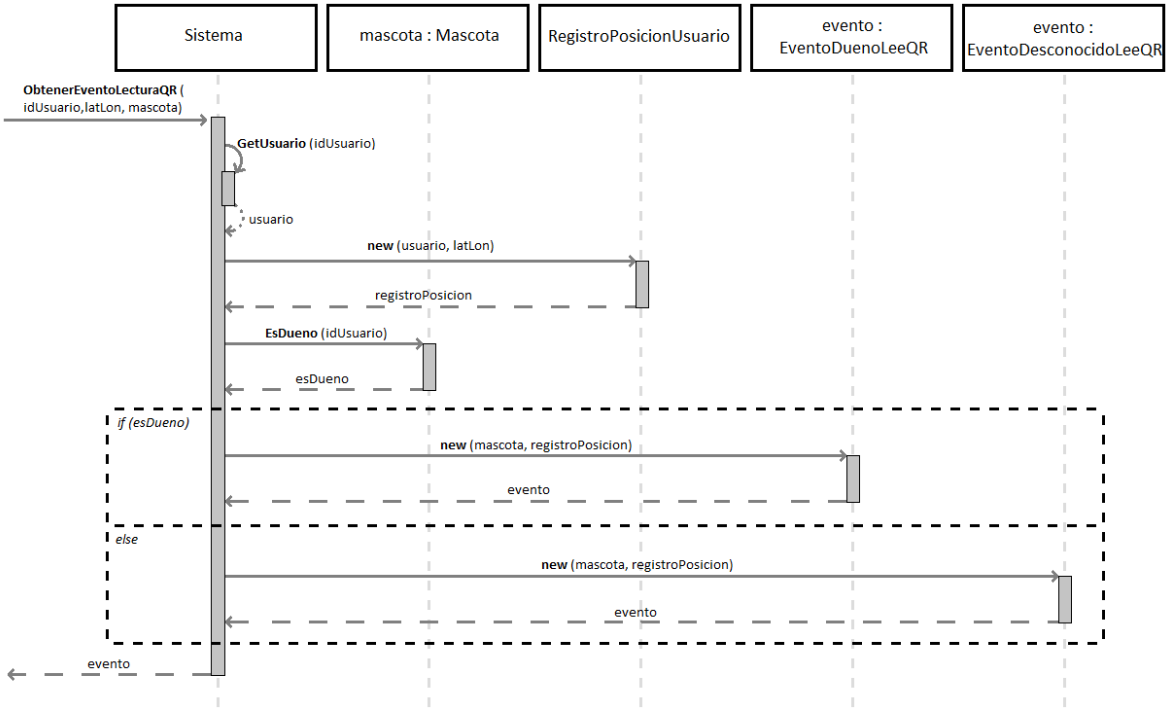


Figura 3.20: Diagrama de secuencia del método `ObtenerEventoLecturaQR()` de Sistema.

A continuación se detallarán los diagramas correspondientes a las interacciones entre eventos y estados. Estos son subsecuencias del diagrama "*Reportar Extravío* " o del diagrama "*Leer código QR de mascota*".

Es importante aclarar que el único lugar en que se instancia el *EventoAnuncioExtravio* es el ya detallado diagrama *Reportar Extravío*. Por lo tanto, en los diagramas siguientes, cuando se vea involucrado el *EventoAnuncioExtravio* debe tenerse en cuenta que se llega a esa secuencia desde el diagrama *Reportar Extravío.*

Por otro lado, el diagrama "*Leer código QR de mascota"* de la Figura 3.19 *(*con su subdiagrama "*Obtención de evento de lectura"* presentado en la Figura 3.20) es el único lugar en que se instancian los eventos *EventoDuenoLeeQR*, *EventoDesconocidoLeeQR*. Por lo tanto, en los diagramas que se presentan a continuación, cuando se vea involucrado alguno de esos dos eventos, debe tenerse en cuenta que se llega a esa secuencia desde el diagrama "*Leer código QR de mascota"*.

- *Reportar Extravío* ➔ *Procesamiento de evento inicial de anuncio de extravío (dueño reporta extravío)*

  En la Figura 3.18 se describió el procesamiento del sistema cuando el dueño de una mascota la reporta como extraviada. Su última operación consistió en una invocación a *ProcesarEvento*, de la instancia de *ReporteExtravio*. Ese método propaga el evento recibido al estado actual del reporte a través del método `Aceptar()` del *EstadoExtravio*. En la Figura 3.21 se detalla esta

interacción entre el estado inicial *EstadoNull* y el *EventoAnuncioExtravio*: Es decir, cuando la mascota se reporta como extraviada sin haber un reporte activo en ese momento.

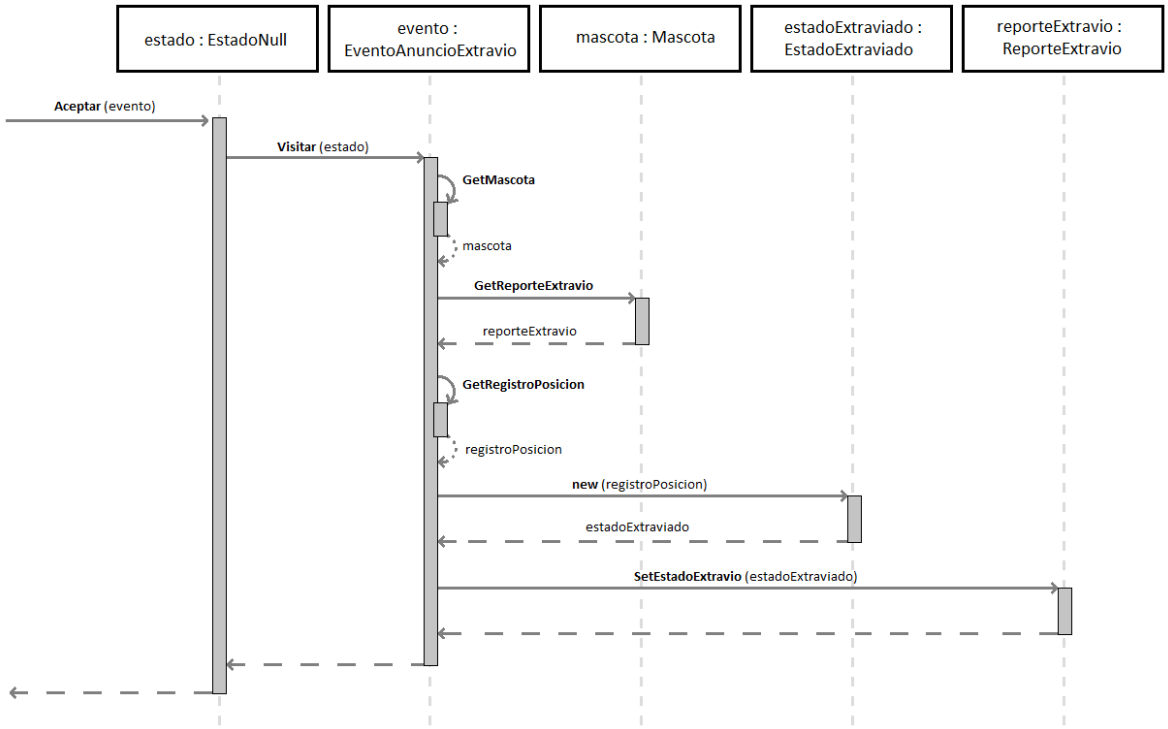


Figura 3.21: Diagrama de secuencia del método `Aceptar()` de *EstadoNull* al ser invocado con un evento de extravío (instancia de la clase *EventoAnuncioExtravio)*.

- *Leer código QR de mascota* ➔ *Subdiagrama: Procesamiento de evento inicial de desconocido leyendo QR*

  Cuando se produce la lectura de un código QR (como se mostró en la Figura 3.19 para todo usuario que lee un código) y se obtiene el evento particular (como se mostró en la Figura 3.20), si la lectura la realiza un desconocido sobre una mascota que no cuenta con un reporte de extravío, se produce la secuencia descripta en la Figura 3.22. Esta figura muestra la interacción entre el estado inicial *EstadoNull* y el *EventoDesconocidoLeeQR*.

  Cabe aclarar que la interacción entre *EstadoNull* y *DueñoLeeQR* no se refleja en un diagrama de secuencia ya que no implica ningún procesamiento. Si la mascota no está extraviada y su dueño la lee, no se producen cambios.

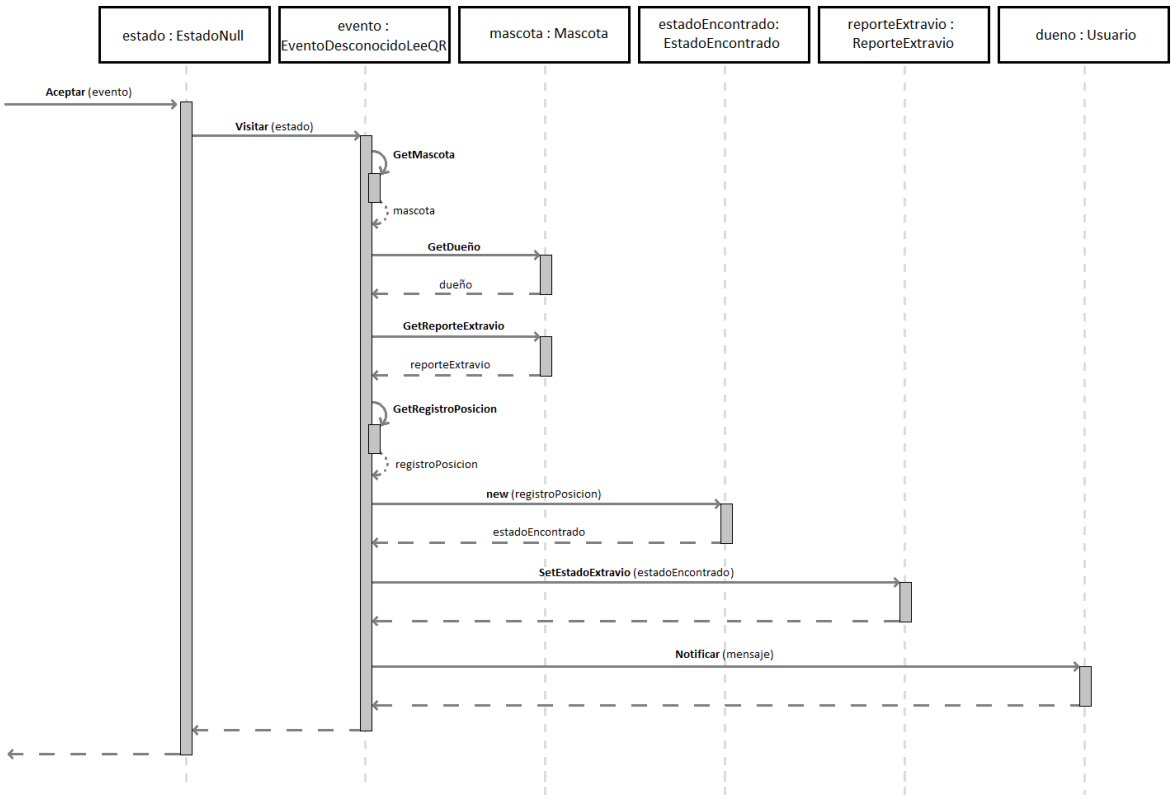


Figura 3.22: Diagrama de secuencia del método `Aceptar()` de EstadoNull al ser invocado con un evento de lectura de desconocido (instancia de la clase EventoDesconocidoLeeQR).

Hasta el momento, las interacciones entre *Eventos* y *Estados* que se describieron involucraron el estado inicial *EstadoNull*, el cual es el estado de inicialización de un reporte de extravío. A partir de ahora se presentarán interacciones que involucran otros estados (como se mostraron en la Figura 3.17).

- *Reportar Extravío* ➔ *Subdiagrama: Procesamiento de evento de anuncio de extravío estando la mascota encontrada.*

  En la Figura 3.23 se presenta la secuencia que se ejecuta cuando el dueño de la mascota anuncia su extravío luego de que la misma fue encontrada por un desconocido: la interacción entre *EstadoEncontrado* y *EventoAnuncioExtravio* descripta se produce (por ejemplo) cuando el dueño de la mascota se entera de su extravío a partir de la alerta que llega porque un desconocido leyó a la mascota. A partir de esta alerta el dueño puede ingresar a la ficha de la mascota, y tras ver que fue encontrada lejos de su casa puede reportarla como extraviada para confirmar su situación (con posibilidad de definir una recompensa) hasta reencontrarse con ella.

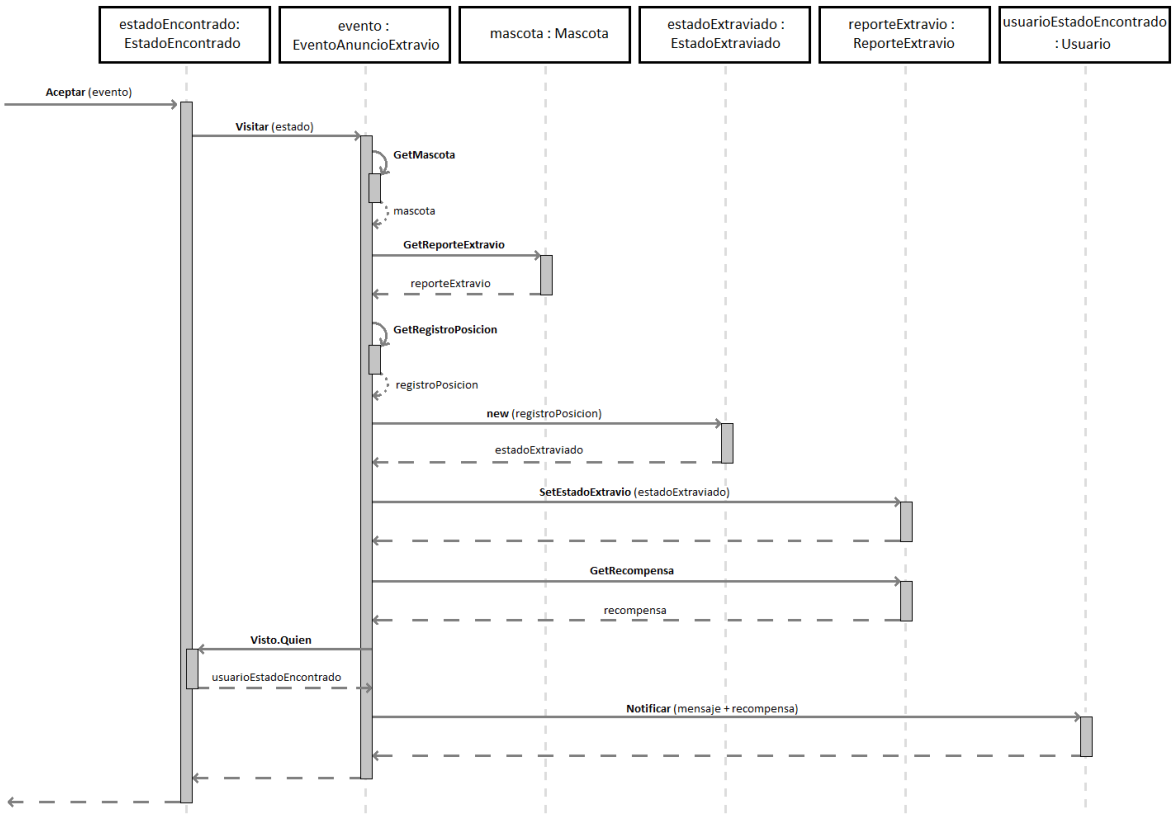


Figura 3.23: Diagrama de secuencia del método `Aceptar()` de *EstadoEncontrado* al ser invocado un evento de extravío (instancia de la clase *EventoAnuncioExtravio*).

- *Leer código QR de mascota* ➔ *Subdiagrama: Procesamiento de evento de desconocido leyendo QR estando la mascota encontrada.*

  Cuando se produce la lectura de un código QR por parte de un desconocido sobre una mascota que ya había sido encontrada por otro usuario se produce la interacción entre el estado inicial *EstadoEncontrado* y *EventoDesconocidoLeeQR*. Esta secuencia es descripta en la Figura 3.24.

  Esta situación podría darse, por ejemplo, cuando una mascota es leída por un desconocido en una posición B minutos después de que otro desconocido haya leído a la mascota en una posición A (por ejemplo, porque la mascota no pudo ser retenida en un lugar). Para el dueño es importante tener acceso a todas las posiciones en las cuales la mascota fue leída por desconocidos aunque esto no implique una transición de estado.

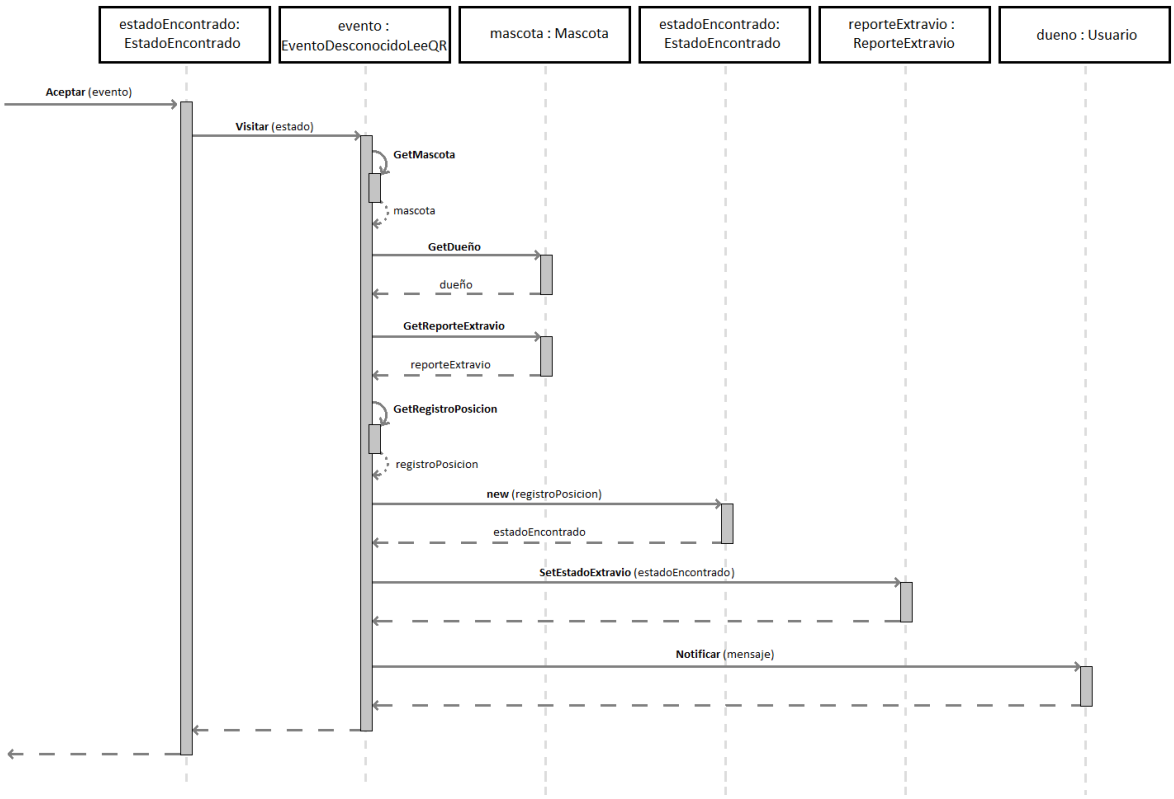


Figura 3.24: Diagrama de secuencia del método `Aceptar()` de *EstadoEncontrado* al ser invocado un evento de lectura de desconocido (instancia de la clase *EventoDesconocidoLeeQR*).

- *Leer código QR de mascota* ➔ *Subdiagrama: Procesamiento de evento de dueño leyendo QR estando la mascota encontrada.*

  Cuando se produce la lectura de un código QR por parte del dueño sobre una mascota que ya había sido encontrada por un usuario se produce la interacción entre *EstadoEncontrado* y *EventoDueñoLeeQR.* Esta secuencia se puede apreciar en la Figura 3.25. Esta situación se produce cuando el dueño lee el código QR de su mascota luego de haber sido encontrada por un desconocido. Es decir, la mascota llegó al *EstadoEncontrado* al haber sido leída por un desconocido y luego, al momento del reencuentro, fue leída por su propio dueño, lo cual dispara el *EventoDueñoLeeQR*. Su procesamiento se puede apreciar en dicha figura.

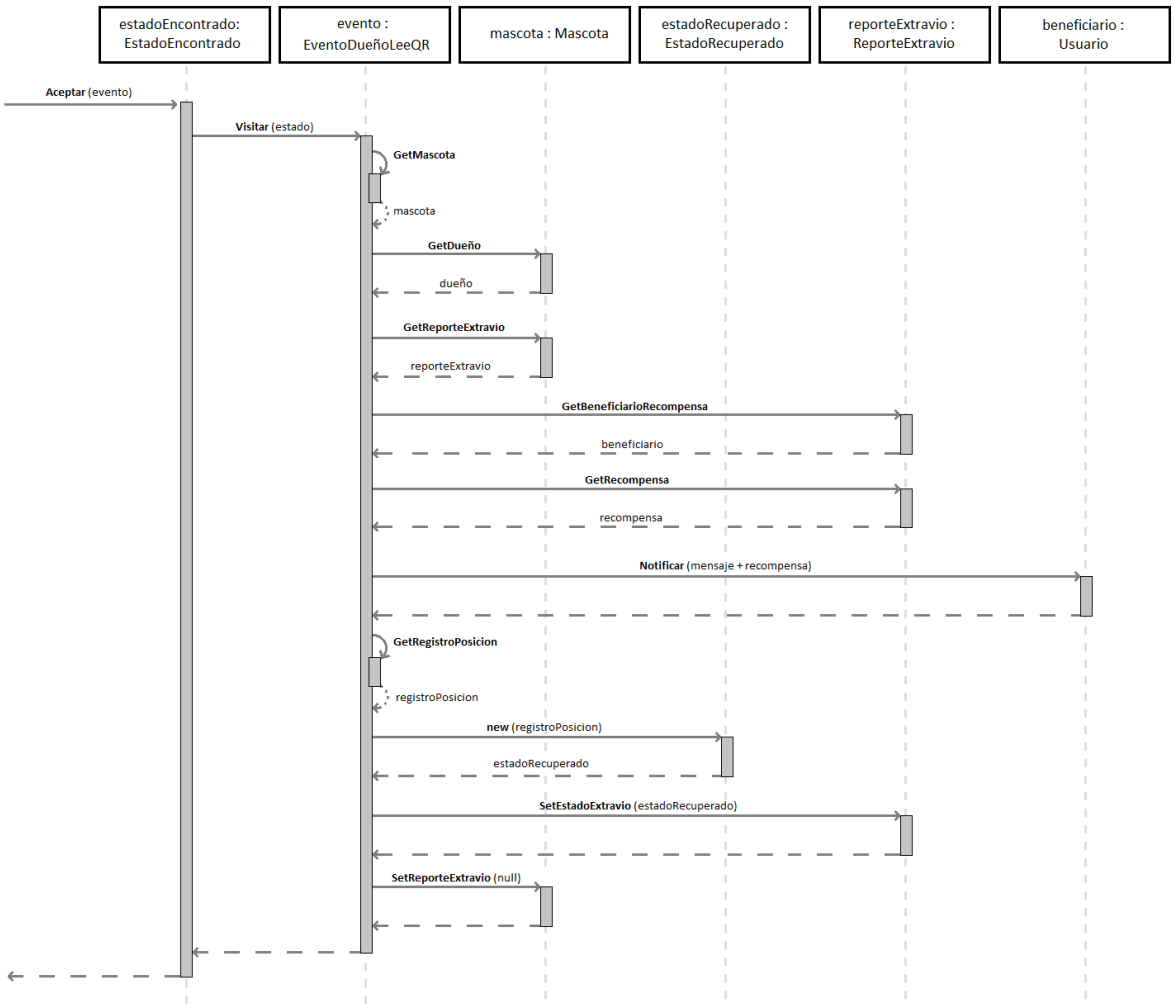


Figura 3.25: Diagrama de secuencia del método `Aceptar()` de *EstadoEncontrado* al ser invocado un evento de lectura del dueño (instancia de la clase *EventoDuenoLeeQR*).

- *Reportar Extravío* ➔ *Subdiagrama: Procesamiento de evento de anuncio de extravío con la mascota ya reportada extraviada.*

  Un dueño de mascota puede reportar a su mascota como extraviada y posteriormente volver a realizar esta operación para (por ejemplo) redefinir la recompensa. Sin embargo esta reiteración en el reporte de extravío no produce ningún cambio de estado.

  En la Figura 3.26 se muestra que la reiteración de anuncio de extravío no requiere mayor procesamiento.

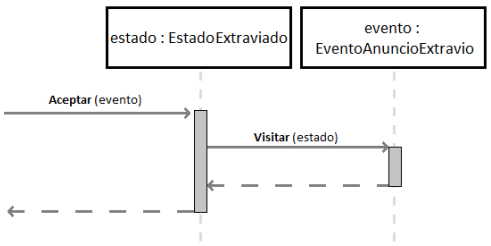


Figura 3.26: Diagrama de secuencia del método `Aceptar()` de *EstadoExtraviado* al ser invocado un evento de extravío (instancia de la clase *EventoAnuncioExtravio*).

- *Leer código QR de mascota ➔ Subdiagrama: Procesamiento de evento de desconocido leyendo QR con la mascota estando extraviada.*

  Cuando un desconocido lee el código QR de una mascota que está reportada como extraviada se produce la interacción entre el *EventoDesconocidoLeeQR* y el *EstadoExtraviado*. Esta interacción se refleja en el diagrama de secuencia de la Figura 3.27. No existe ninguna diferencia entre procesamiento del *EventoDesconocidoLeeQR* cuando el estado actual es *EstadoExtraviado* (Figura 3.27) y cuando el estado actual es *EstadoNull* (Figura 3.22). Pero ambos escenarios tienen orígenes diferentes ya que la forma de llegar al *EstadoExtraviado* es con el dueño reportando el extravío; mientras que el *EstadoNull* es el estado de inicialización de un reporte, lo cual implica que no existía reporte en el momento en que el evento se disparó.

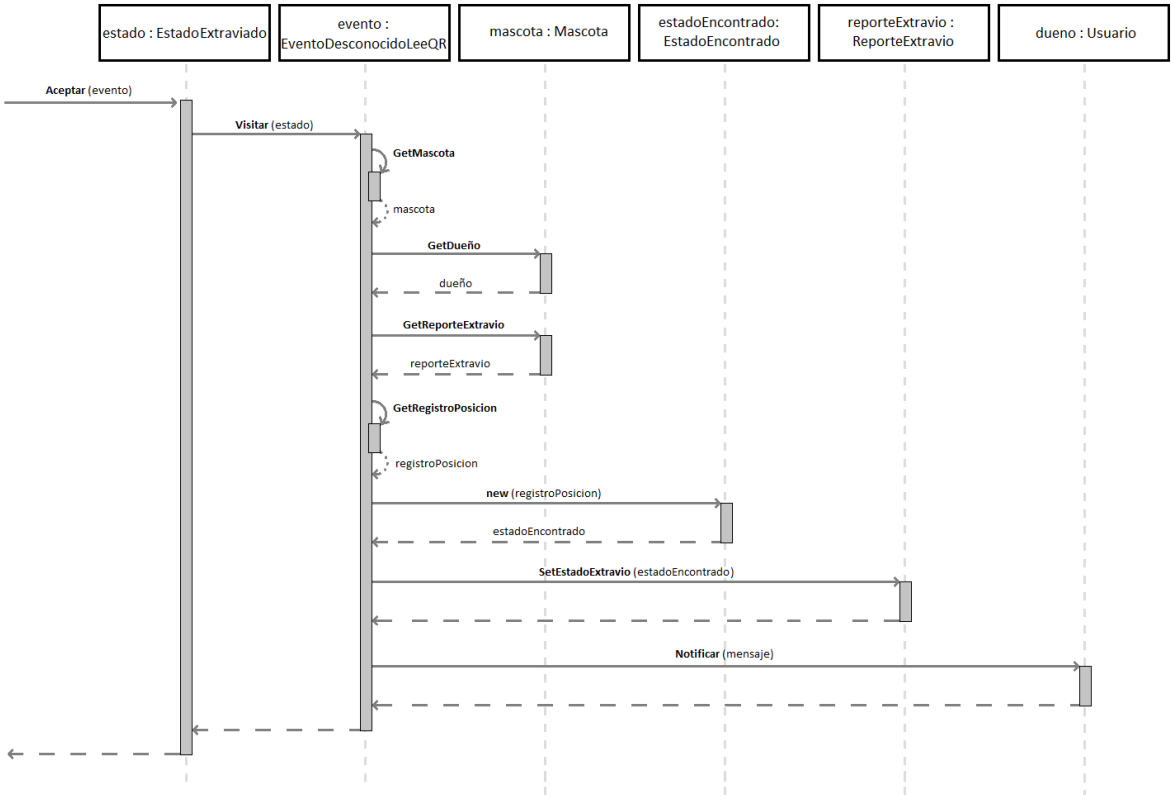


Figura 3.27: Diagrama de secuencia del método `Aceptar()` de *EstadoExtraviado* al ser invocado un evento de lectura de desconocido (instancia de la clase *EventoDesconocidoLeeQR*).

- *Leer código QR de mascota ➔ Subdiagrama: Procesamiento de evento de dueño leyendo QR con la mascota estando extraviada.*

  Cuando el dueño de la mascota lee su código QR estando la misma reportada como extraviada se produce la interacción entre el *EventoDueñoLeeQR* y el *EstadoExtraviado*. Existe una única forma de disparar el *EventoDueñoLeeQR* pero dos formas de llegar al *EstadoExtraviado*:

  - Que el dueño advierta por su cuenta que su mascota se extravió y la reporte como extraviada (como se mostró en la Figura 3.21).
  - Que el dueño, sin haber el advertido el extravío, reciba una notificación automática producida por la lectura de su mascota por parte de un desconocido (ver Figura 3.22) y tras esto, la reporte como extraviada (ver Figura 3.23).

  Independientemente de cuál de estas dos formas derivó en el *EstadoExtraviado*, la posterior lectura del código QR por parte del dueño de la mascota (al momento del reencuentro) deriva en la interacción entre el *EventoDueñoLeeQR* y el *EstadoExtraviado* como se muestra en la Figura 3.28.

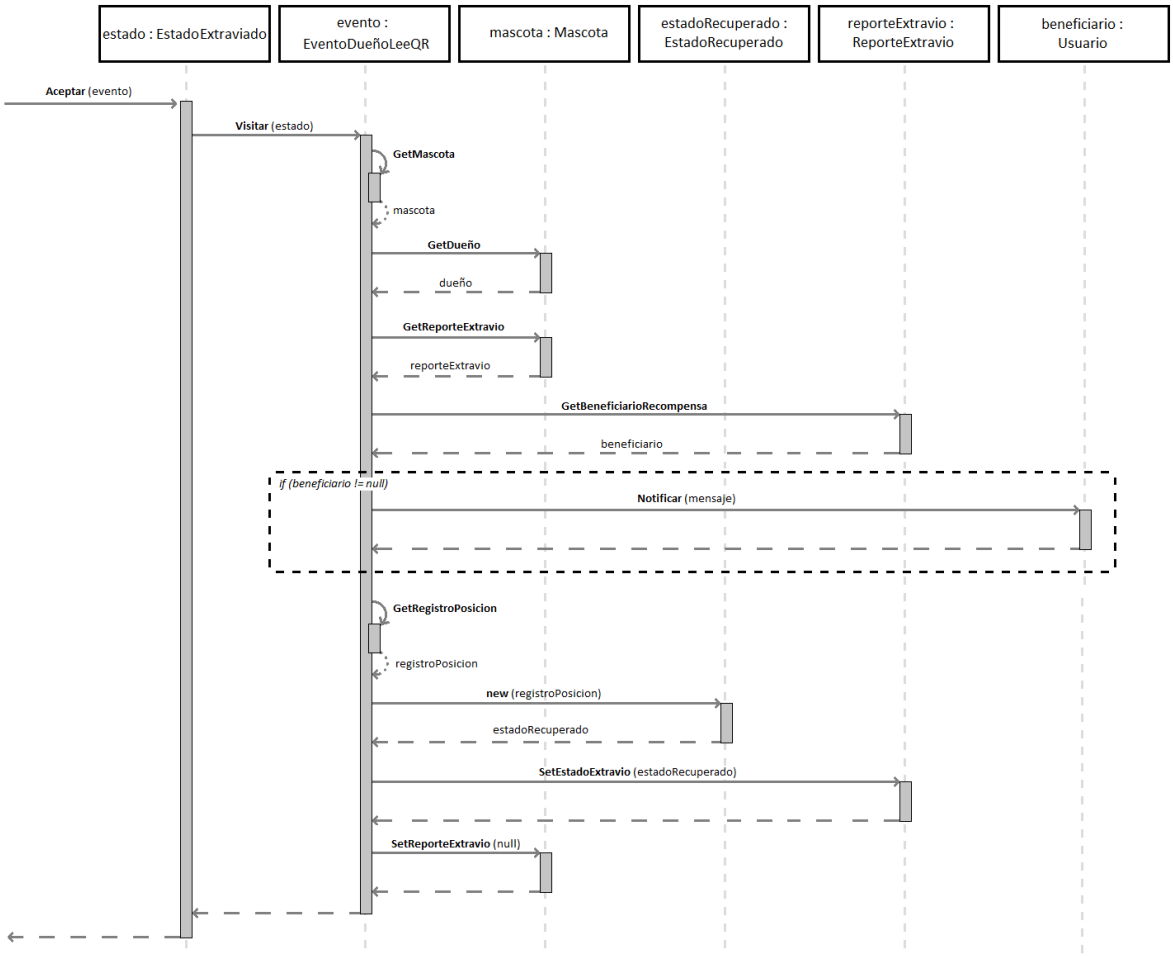


Figura 3.28: Diagrama de secuencia del método `Aceptar()` de *EstadoExtraviado* al ser invocado un evento de lectura del dueño (instancia de la clase *EventoDuenoLeeQR*).

Capítulo 4. Prototipo desarrollado

El prototipo fue planteado como una Aplicación Web, que puede ser utilizada tanto desde dispositivos móviles como desde PCs por su condición de responsive[34] (que se adapta a las dimensiones del browser). A continuación se describen todas las herramientas o APIs usadas para el desarrollo del mismo, describiendo para qué fue utilizada cada una de ellas.

- El prototipo fue implementado con el lenguaje C# de la plataforma .NET[35] y utilizando el entorno de desarrollo Microsoft Visual Studio 2013[36].
- Para el desarrollo web se utilizó el framework MVC.NET 3[37], el cual implementa el patrón de diseño MVC (model-view-controller).
- Para determinar la posición de los usuarios se utilizó la API de posicionamiento de HTML5 (*HTML5 Geolocation API*[38]). Esta API se ajusta a los mecanismos de posicionamiento disponibles para cada dispositivo móvil (los cuales se detallaron, por ejemplo, en la Tabla 2.3) y acorde a esto, los combina para determina la posición de la manera más precisa posible.
- Se utilizó el servicio *Twilio*[39] para enviar las notificaciones por SMS.
- También se utilizó la *API de Google Maps*[40] para presentarle al usuario las ubicaciones de mascotas, recursos y necesidades.
- Se optó por utilizar en este prototipo los códigos QR. Para construirlos se utilizó una librería con licencia GNU llamada QRCodeLib[41]. Esta librería con licencia GNU se utilizó para generar los códigos QR, donde su input está conformado por un string (la URL de la ficha de la mascota), y parámetros de configuración (tamaño y nivel de corrección, tal como se describió en el apartado "QR Code" del Capítulo 2).

4.1. Estructura de la solución desarrollada.

En el contexto de *Visual Studio* (IDE de la plataforma .NET) cuando se crea una aplicación, un sitio web, una aplicación web, un script, un complemento, etc. se debe comenzar con un *Proyecto*. En un sentido lógico, un *Proyecto* contiene todos los archivos de código fuente, iconos, imágenes, archivos de datos y cualquier otra cosa que se compilará en un programa ejecutable o sitio web, etc. necesarios para realizar la compilación.

Un *Proyecto* también contiene toda la configuración del compilador y otros archivos de configuración que podrían ser necesarios en diversos servicios o componentes con los que el

[34] Más información sobre *Web Responsive Design*: http://www.w3schools.com/html/html_responsive.asp (Último acceso: 12-dic-2016)
[35] https://www.microsoft.com/net. (Último acceso: 12-dic-2016).
[36] https://www.microsoft.com/es-ar/download/details.aspx?id=44914. (Último acceso: 12-dic-2016).
[37] https://www.asp.net/mvc/mvc3. (Último acceso: 12-dic-2016).
[38] http://www.w3schools.com/html/html5_geolocation.asp (Último acceso: 12-dic-2016).
[39] https://www.twilio.com/. (Último acceso: 12-dic-2016).
[40] https://developers.google.com/maps/?hl=es-419. (Último acceso: 12-dic-2016).
[41] https://www.nuget.org/packages/ThoughtWorks.QRCode/. (Último acceso: 12-dic-2016).

programa se comunicará. Un *Proyecto* se encuentra, en un sentido lógico y en el sistema de archivos, en una *Solución* que puede contener uno o más *Proyectos*, junto con información de compilación, la configuración de ventana de *Visual Studio* y archivos varios que no estén asociados a ningún *Proyecto*.

En el caso del prototipo que forma parte de esta Tesina, la *Solución* (llamada *GeolocalizacionQR*) está conformada por tres *Proyectos*, y la forma en que se presentan en *Visual Studio* se puede ver en la Figura 4.1

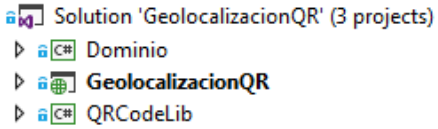


Figura 4.1: Estructura de la solución tal como se presenta en Visual Studio

Veamos más nivel de detalle de los proyectos descriptos en la Figura 4.1 (se hará hincapié sólo en los proyectos *Dominio* y *GeolocalizacionQR*, dado que el proyecto *QRCodeLib* contiene lo relacionado a dicha librería como se mencionó anteriormente). Las responsabilidades y alcance de cada uno de estos proyectos se describen a continuación:

- *Dominio*

  Es el proyecto más relevante del prototipo ya que contiene el modelo tal como se lo describió en el Capítulo 3, incluyendo la definición de sus clases y su comportamiento. La estructura interna de este proyecto es simple, tal como se puede observar en la Figura 4.2.

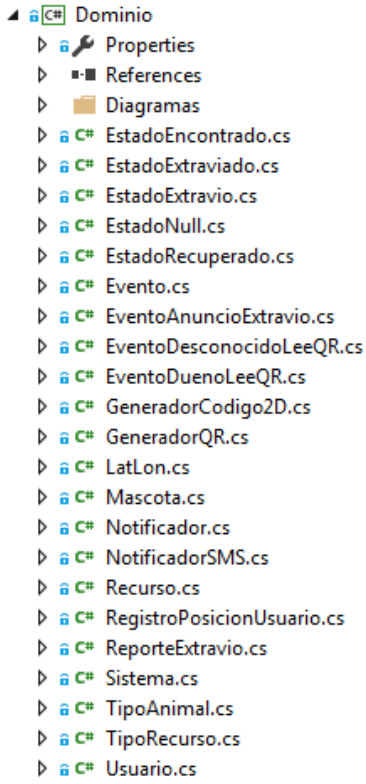


Figura 4.2: Estructura del proyecto Dominio.

En la carpeta *Diagramas* (Figura 4.2) pueden encontrarse los diagramas de clases desarrollados para esta Tesina.

- *GeolocalizacionQR*

  Es la aplicación web que determina la interacción con el usuario. Cabe destacar que si el desarrollador determinara conveniente implementar una aplicación mobile, podría desarrollarla como un proyecto paralelo a *GeolocalizacionQR* sin necesidad de tocar una sola línea del proyecto *Dominio*. En la Figura 4.3 se puede apreciar cómo está compuesto el proyecto *GeolocalizacionQR.*

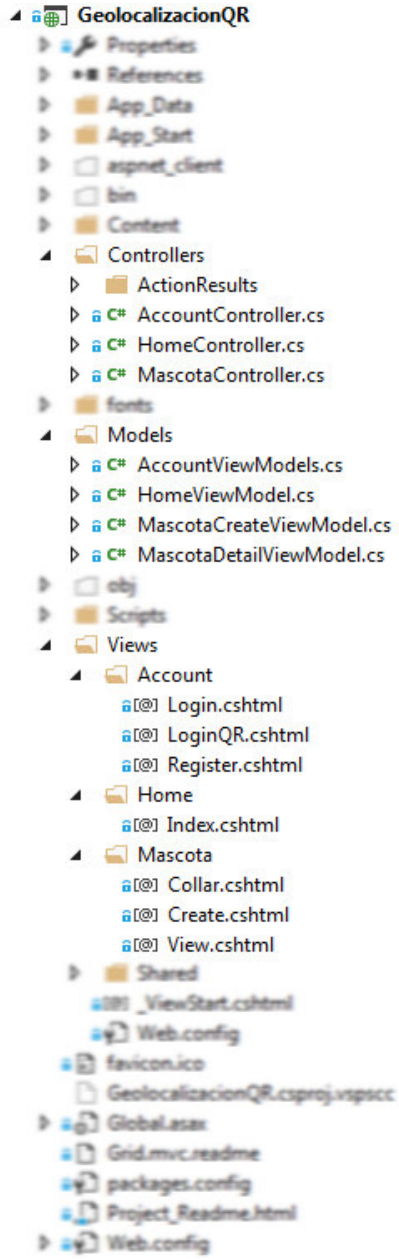


Figura 4.3: Estructura del proyecto GeolocalizacionQR.

Como se mencionó anteriormente, la aplicación web cumple con el patrón *Model-View-Controller*, y a continuación se hará foco en esos tres componentes, ya que existen otros archivos y carpetas de configuración propios de las herramientas utilizadas, cuyo análisis no se considera relevante para la presente tesina. Veamos a continuación más nivel de detalle de la Figura 4.3 en la que se destacan los principales elementos:

- La carpeta *Controllers* contiene:

  AccountController: Contiene el conjunto de acciones (puntos de entrada) relativas a la autenticación, registración y salida de la aplicación por parte del usuario. Por ejemplo, los códigos QR generados por el prototipo referencian a la acción "*LoginQR*" del *AccountController*, que se encargará de solicitar al usuario los datos de autenticación.

  HomeController: Contiene el conjunto de acciones relativas a la carga de la *home page*. Contiene una acción denominada "*Index*" que obtiene (entre otras cosas) las mascotas a presentar en la *home page*.

  MascotaController: Contiene las acciones correspondientes a la interacción con mascotas individuales, como ser su alta en el sistema, visualización de detalles de una mascota, generación de identificación QR, o reporte de extravío. Estas acciones interactúan directamente con la fachada del modelo: la clase Sistema.

- La carpeta *Models* contiene:

  AccountViewModels: Contiene los models relacionados con la funcionalidad de autenticación, incluyendo modelo para login y para registración.

  HomeViewModel: Es el model utilizado para renderizar la *home page*, y contiene el listado de Mascotas como así también el Usuario autenticado (en caso de estarlo).

  MascotaCreateViewModel: Es el model utilizado para registrar el alta de una mascota.

  MascotaDetailViewModel: Es el model utilizado para cargar la vista de detalles de una mascota.

  Cabe aclarar que estos models no cuentan con comportamiento, sino que son contenedores de datos que se cargan en los controllers y se presentan en las vistas (ejemplo: consulta de datos); o bien que el usuario carga en las vistas y viajan a los controllers (ejemplo: alta de datos).

- La carpeta *Views* contiene:

  *Account**Login*: Es el formulario de autenticación de usuario.

  *Account**LoginQR*: Es el formulario de autenticación referenciado por los códigos QR. A diferencia del anterior, implica un redireccionamiento automático (tras pasar la autenticación) hacia la ficha de una mascota particular (con el consecuente procesamiento de envío de notificaciones y cambios de estado por lectura de código QR).

*Account**Register*: Es el formulario de registración de usuarios.

*Home**Index*: Es la *home page*.

*Mascota**Collar*: Es la vista en la cual se presenta la etiqueta identificatoria de una mascota, lista para imprimir.

*Mascota**Create*: Es el formulario de alta de mascota.

*Mascota**View*: Es la vista de detalles de mascota.

4.2. Dificultades y decisiones de implementación

Durante el desarrollo del prototipo debieron afrontarse diferentes dificultades técnicas propias de la implementación, que debieron ser resueltas mediante decisiones de implementación.

- *Accesibilidad desde internet*

  Durante las primeras etapas del desarrollo fue posible trabajar ejecutando la aplicación localmente (http://localhost), lo cual permitió desarrollar los formularios, como así también realizar tests de caja negra de todos los componentes del modelo.

  Posteriormente, dada la necesidad de acceder desde dispositivos móviles para (en principio) confirmar la correcta visualización de la aplicación, fue necesario hacerla accesible desde cualquier dispositivo en la red LAN (http://192.168.0.13). Esto requirió la apertura del puerto asociado a la aplicación en el firewall del sistema operativo.

  Una vez que se logró una versión completamente funcional del prototipo, se dio lugar al test funcional, pero al tratarse de una aplicación que involucra *posicionamiento*, el hecho de estar limitado a trabajar en la red LAN representaba una restricción que dificultaba las pruebas reales del prototipo, al no contar con la posibilidad de interactuar con el sistema desde dispositivos móviles a distancias considerables que pudiesen ser reflejadas en los mapas del historial de una mascota.

  La accesibilidad del prototipo desde internet se logró mediante una configuración de *Port Forwarding* en el router. El router tiene una dirección IP asignada por el proveedor de internet, y el *Port Forwarding* permite redireccionar los requests recibidos por el router en determinado puerto hacia una <IP>:<puerto> de la red LAN. Por lo tanto, mi prototipo es accesible desde internet utilizando la IP de mi router y un número de puerto en particular. Como consecuencia de esto, los códigos QR generados por mi prototipo (los cuales contienen una URL hacia una ficha de mascota) también deben referenciar al <IP>:<puerto> correspondiente.

- *Seguridad con HTTPS*

  Desde el momento en que el prototipo fue publicado en la red LAN se observó que al ingresar a la aplicación por IP de LAN (y no por *localhost*), el prototipo dejaba de obtener los datos de *Latitud* y *Longitud* requeridos para presentar los mapas. Estos datos son obtenidos mediante la *HTML5 Geolocation API* (API de posicionamiento de HTML5). Tras investigar la causa de este problema se identificó que la tendencia de los browsers, por cuestiones de seguridad, es que la posición del usuario solo puede ser obtenida desde sitios HTTPS (o bien *localhost*).

  Por este motivo, dado que el prototipo requería ser utilizado desde internet, fue necesario publicarlo como HTTPS. Se identificaron dos formas de lograr esto:

  I. Tramitando un certificado (pfx) autorizado.
  II. Construyendo un certificado Self-signed e instalándolo en los dispositivos desde los cuales se harían las pruebas.

  Se optó por la segunda alternativa, lo cual implicó seguir un complejo procedimiento que involucró la ejecución de un script para powershell de Windows (denominado New-SelfSignedCertificateEx.ps1[42]) tras aplicarle algunas correcciones.

- Envío de SMS

  Se decidió que el prototipo enviara las alertas a través de SMS. Se identificó una plataforma de *web services* denominada Twilio[43], donde uno de ellos se utiliza para enviar SMS. Al crear una cuenta en el sistema, el usuario gestiona un *AccountSid*, un *AuthToken*, y un número de teléfono (*From*), utilizados para identificar la cuenta al momento de la invocación del web service. Cada nuevo usuario cuenta con una precarga de saldo para pruebas y evaluación. Este saldo es escaso pero suficiente para evaluar el desempeño del *web service*. En etapas avanzadas de desarrollo y testing resultó necesario aumentar el saldo para continuar utilizando el servicio.

  La clase *NotificadorSMS* del *Dominio* es la que tiene definida el *AccountSid*, *AuthToken* y *From*. Para contar con el mecanismo de notificación desarrollado en el prototipo será necesario, por tanto, crear una cuenta en Twilio, gestionar *AccountSid*, *AuthToken* y *From* y cargar dicha configuración en *NotificadorSMS*.

[42] https://gallery.technet.microsoft.com/scriptcenter/Self-signed-certificate-5920a7c6/view/Discussions
[43] https://www.twilio.com/

Capítulo 5. Casos de prueba del prototipo desarrollado

En este capítulo se mostrará usando un ejemplo hipotético el funcionamiento del prototipo presentado en el Capítulo 4. Para este ejemplo, se cargó información de algunas mascotas con el fin de poder realizar esta simulación, para así mostrar la funcionalidad del prototipo.

Cuando un usuario no identificado ingresa a la "*home page*" de la aplicación se le presenta únicamente el listado de mascotas actualmente extraviadas. Este listado (como los demás que conforman la aplicación) presenta opciones básicas de filtrado y ordenamiento. Esto se representa en la Figura 5.1[44]. Es decir, cualquier usuario más allá de estar o no registrado en el sistema puede ver esta lista.

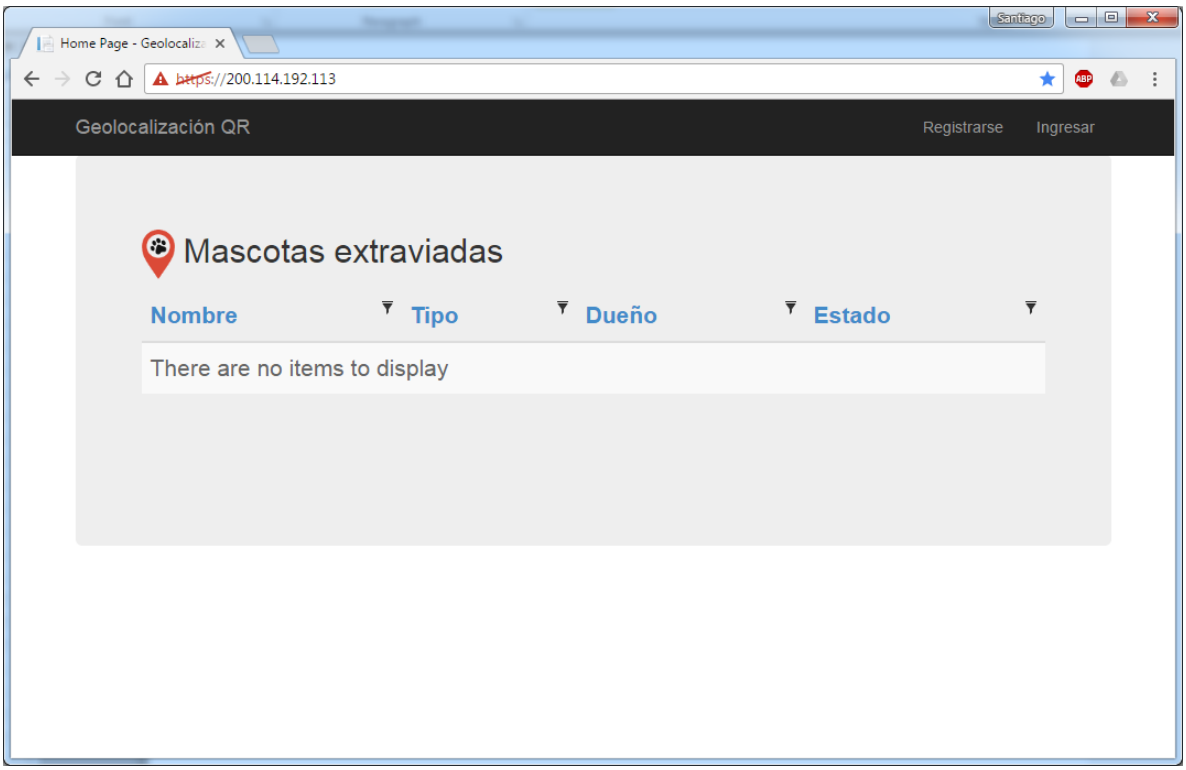


Figura 5.1: *Home page* del prototipo vista por un usuario no identificado.

Al ingresar desde un *Smartphone* se tiene una vista similar a la antes mostrada (en la Figura 5.1), como se puede apreciar en la Figura 5.2, donde se muestra el browser apaisado para una mejor visualización de la información. Se puede apreciar que la interfaz se adapta a la dimensiones de la pantalla del *Smartphone* (las captura del *smartphone* se realizaron usando un *iPhone 6s*).

[44] Se puede apreciar en las capturas de pantallas de todo el capítulo que el *https* aparece en rojo, esto se debe a que el certificado que se está usando, al ser autofirmado no es confiable para el browser por no estar firmado por una Autoridad de Certificación. El certificado autofirmado permite al prototipo ser accesible por https (con la encriptación que esto habilita), y esto es necesario para poder obtener la posición con HTML5, como se mencionó en la Sección 4.2.

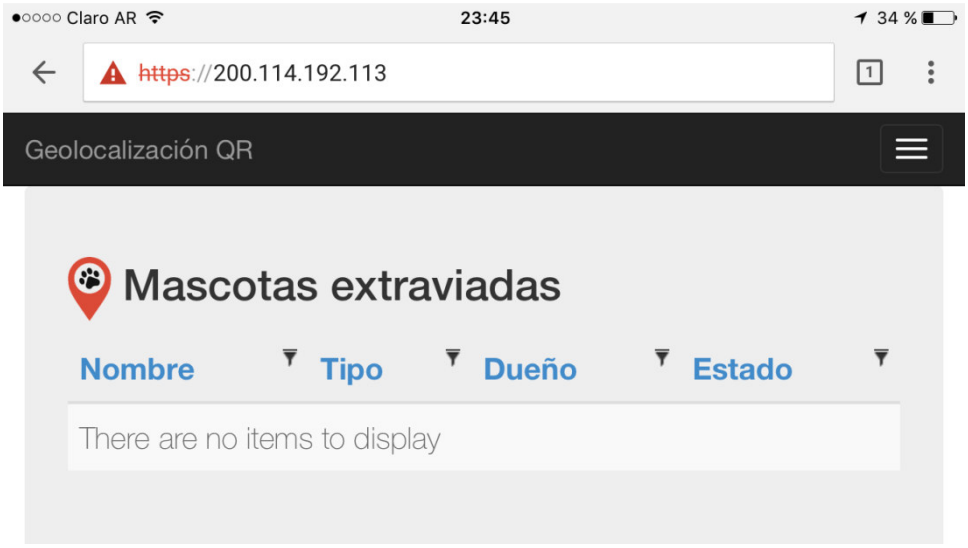


Figura 5.2: Home page del prototipo vista desde un Smartphone por un usuario no identificado.

Los usuarios registrados pueden identificarse en el sistema, ingresando su email y password. El formulario de ingreso se puede visualizar en las Figura 5.3.

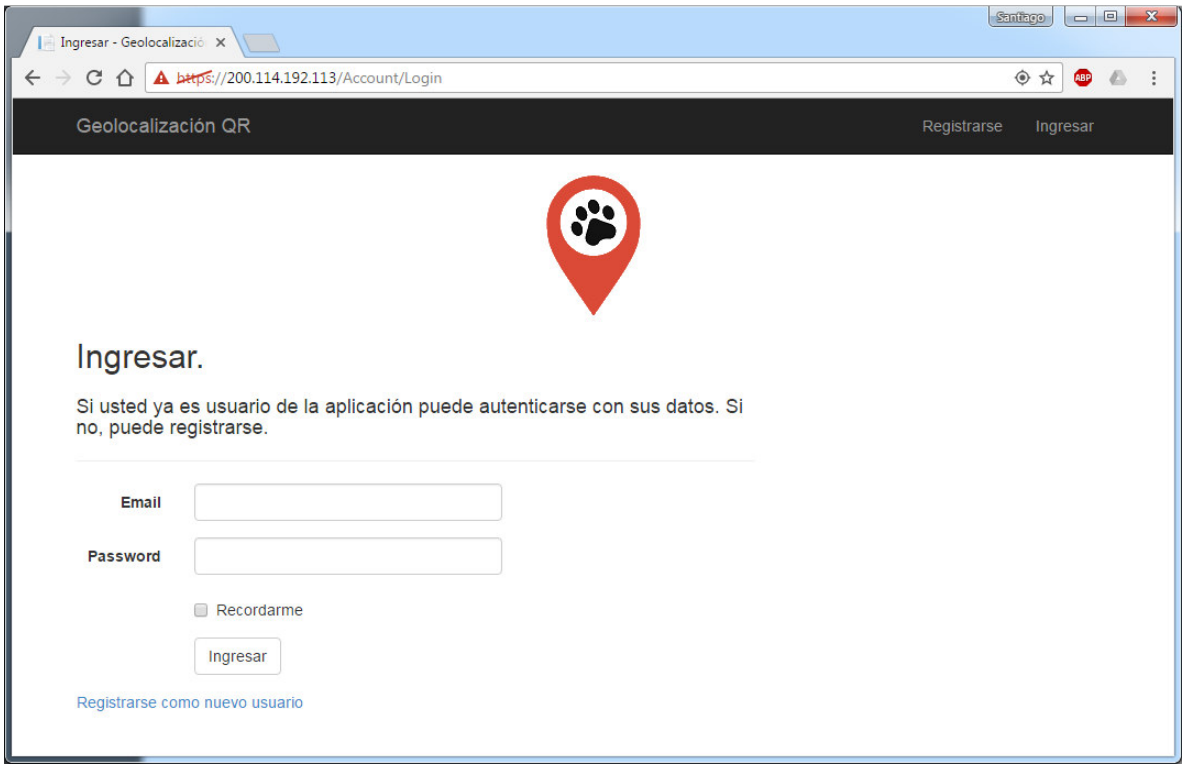


Figura 5.3: Formulario de autenticación.

Los usuarios no registrados pueden registrarse en la aplicación, mediante el formulario presentado en la Figura 5.4.

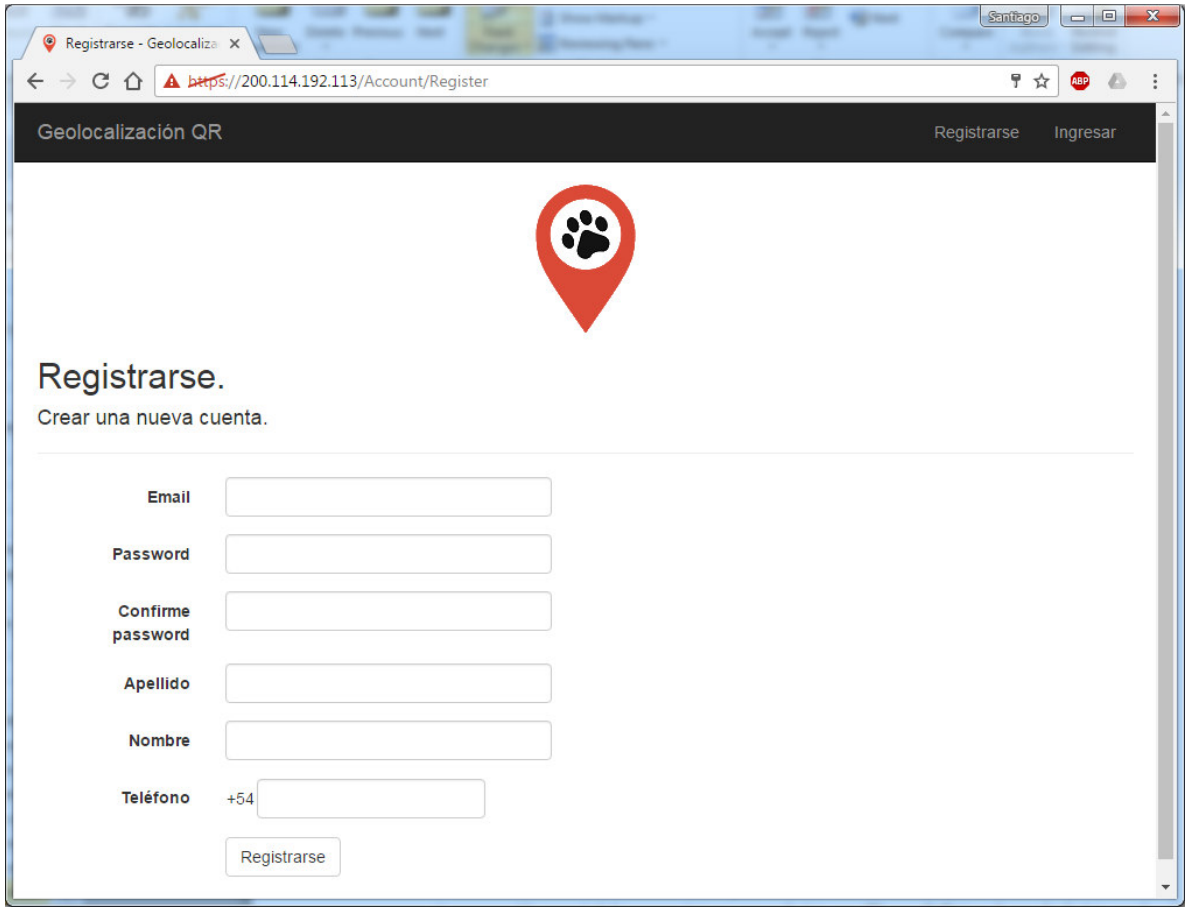


Figura 5.4: Formulario de registración.

Cuando una persona lee un código QR de una mascota, lo primero que presenta el sistema es el formulario de autenticación presentado en la Figura 5.3. Es probable que cuando una persona encuentre a una mascota portando un código QR desconozca la existencia del sistema, por lo cual su primera acción consistiría en registrarse en el sistema (Figura 5.4) tras lo cual el sistema le presentará la ficha de la mascota a través de la cual ingresó al sistema (como se verá más adelante en la Figura 5.6).

Para usuarios identificados, en la página principal se incorporan tres grillas:

- *Mis mascotas*: Mascotas de las cuales el usuario autenticado es dueño.
- *Mascotas que encontré*: Aquellas mascotas que, habiendo estado o estando extraviadas, fueron encontradas por el usuario.
- *Mascotas en el sistema*: Todas las mascotas registradas en el sistema.

Estas grillas se pueden apreciar en la Figura 5.5.

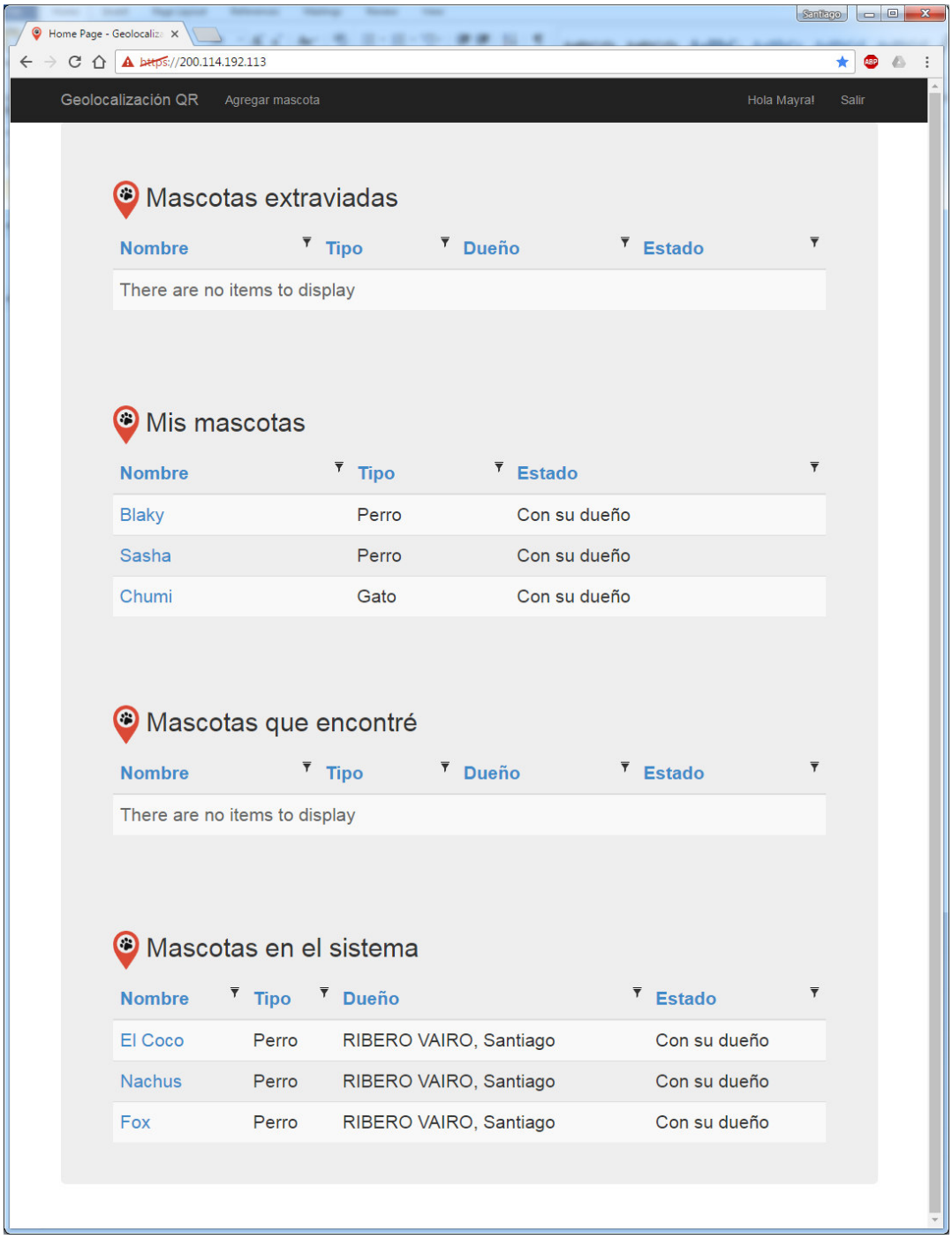


Figura 5.5: Visión de la home page de un usuario autenticado.

El usuario puede ingresar a las fichas de las mascotas. Al ingresar a la ficha de una mascota propia se presenta una botonera de acciones:

- *Imprimir identificación*
- *Reportar Extravío*

Y los datos de la mascota. La información de contacto es visible solo para aquellos usuarios que interactuaron directamente con la mascota (es decir: *leyeron su código QR*) para evitar que usuarios desconocidos molesten al dueño, o éste quede expuesto a estafas. Lo antes descripto se puede visualizar en la Figura 5.6.

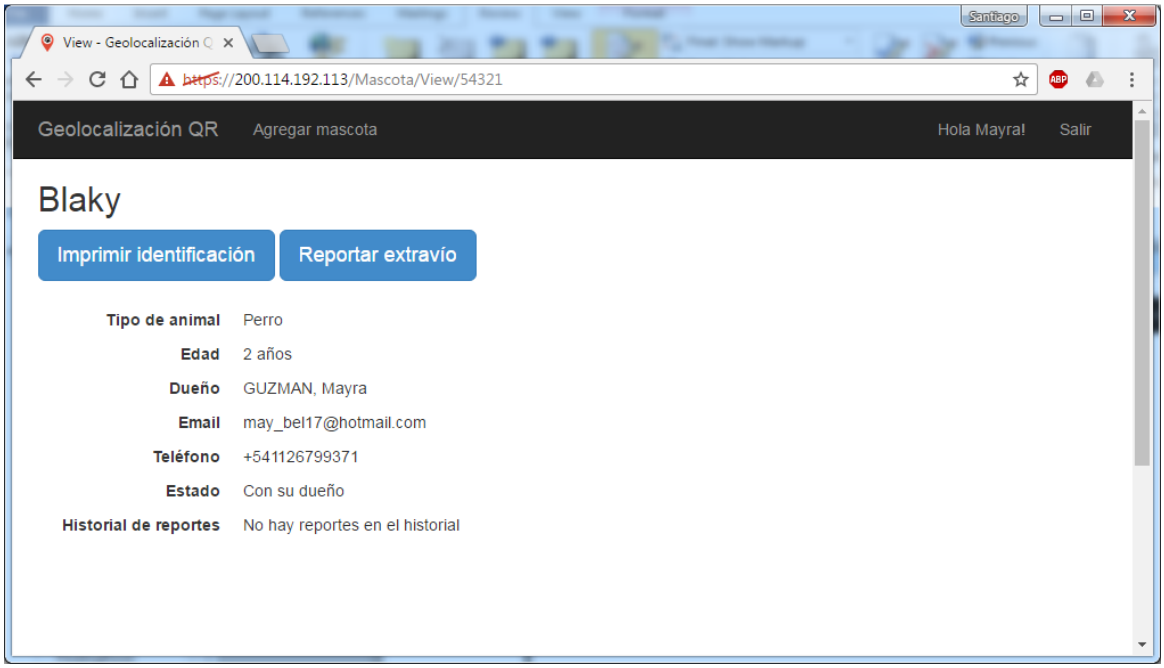


Figura 5.6: Vista del formulario de detalles de una mascota por parte de su dueño.

Al ingresar a "*Imprimir identificación*" (de la Figura 5.6) se presenta en pantalla la imagen presentada en la Figura 5.7 (con QR y datos de mascota). La intención es que el usuario la imprima, la recorte, remueva los rectángulos blancos y pliegue la etiqueta por la línea media (ver Figura 5.8). Esto le permitirá insertar la etiqueta en el collar a través de los dos pares de agujeros superpuestos (ver Figura 5.9). De esta manera, el collar quedará mostrando el código QR, y (complementariamente) los datos de contacto quedarán del lado de adentro.

A los fines prácticos puede ser conveniente imprimir la etiqueta en un papel resistente e impermeable, o bien incorporar la etiqueta en un protector plástico.

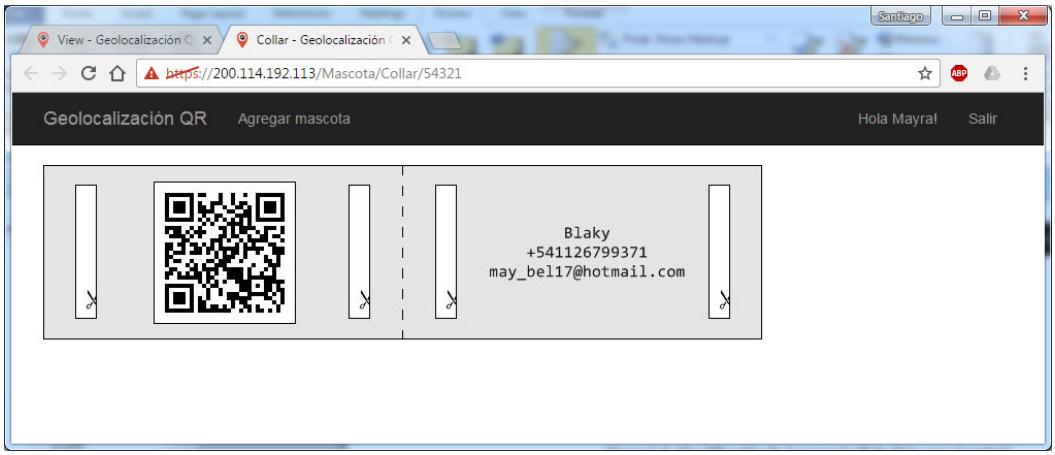


Figura 5.7: Identificación de la mascota Blaky lista para imprimir.

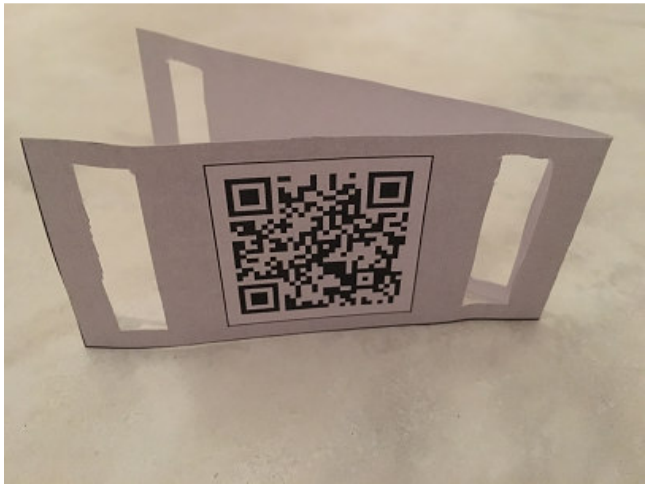


Figura 5.8: Identificación recortada y plegada.

Figura 5.9: Identificación enhebrada en el collar.

El dueño también puede "*Reportar extravío*" (opción anteriormente mencionada cuando se mostró la Figura 5.6), lo cual despliega el input para definir la recompensa (de existir), como se puede apreciar en la Figura 5.10. Esta acción hace que la mascota pase a presentarse en la grilla de mascotas extraviadas (cuando las mismas se listan en la *home page*).

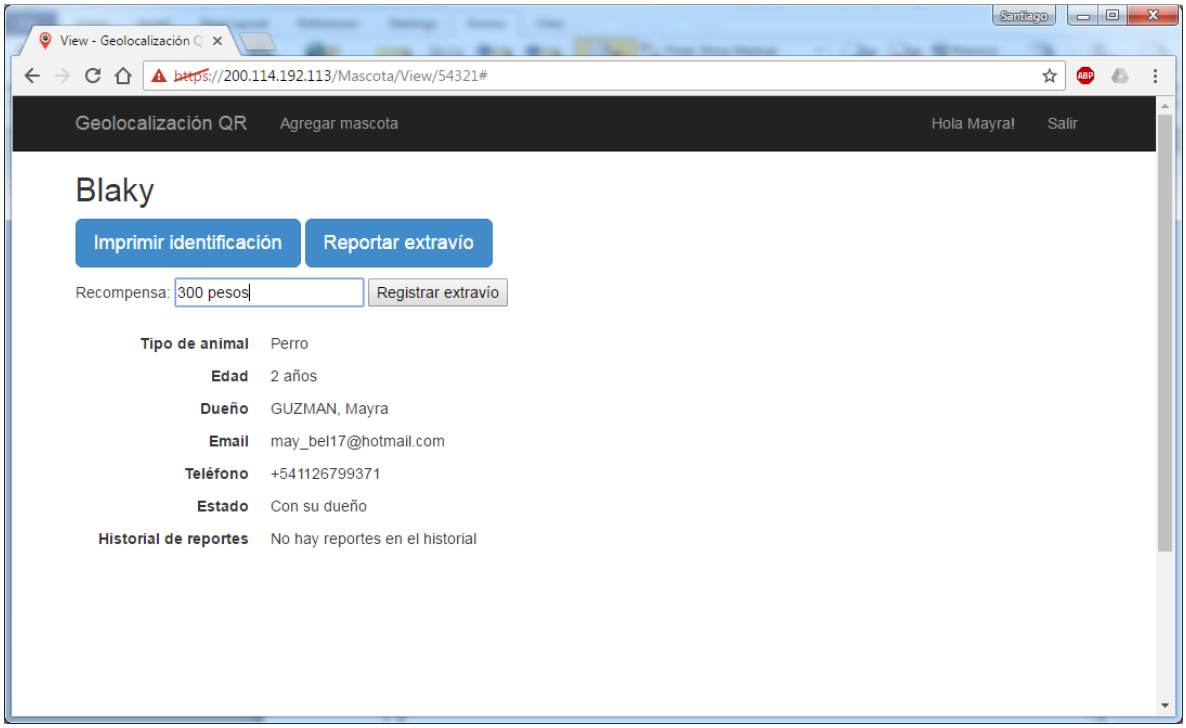


Figura 5.10: Usuario registrando el extravío de su mascota Blaky.

La vista de detalles de la mascota pasa a presentar un *Historial de Reportes* (en la Figura 5.10 se puede apreciar vacía aun), incluyendo una grilla con los estados por los que pasó la mascota (de más antiguo a más reciente) y al hacer clic en cada ícono de la columna *Dónde*, el mapa presenta la posición en que tuvo lugar ese evento.

En la Figura 5.11 se puede apreciar los detalles de un extravío, donde se indica quién lo reporto y en el mapa se puede visualizar dónde se produjo dicho extravío.

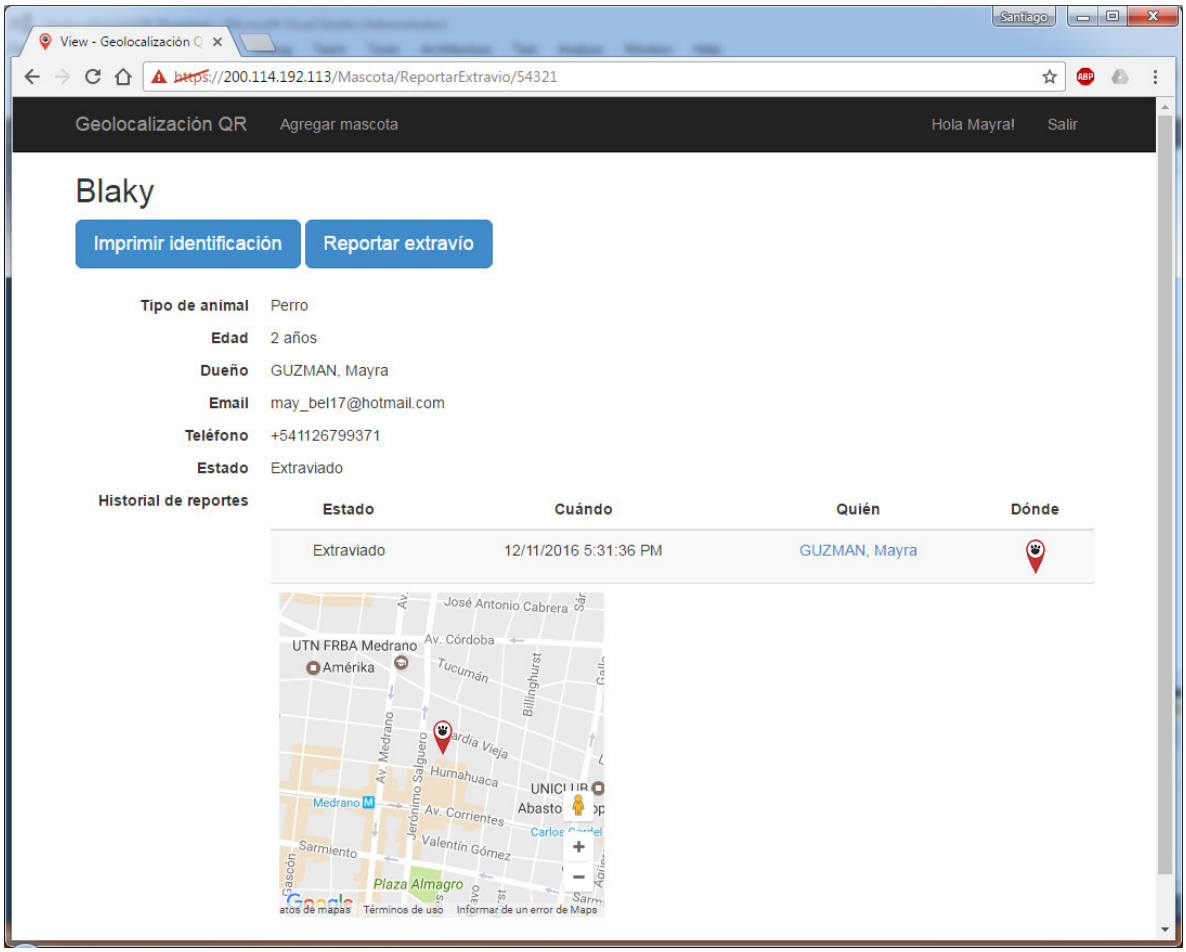


Figura 5.11: Vista de la ficha de la mascota Blaky por parte de su dueño luego de haber sido reportada como extraviada.

Cuando otro usuario encuentra a la mascota y lee su código QR, tras completar el formulario de autenticación (o de registrarse, en caso de no tener cuenta en el sistema), se presenta la ficha de la mascota que encontró, incluyendo el historial de estados (donde para cada estado puede desplegarse el mapa de dónde se dio). Esto se puede visualizar en la Figura 5.12, donde se aprecia cómo el estado de la mascota fue variando. Cada evento sucedió en lugares diferentes y para ver la posición de los mismos basta con desplegar el icono de la columna “Donde” correspondiente al evento. En esta figura se muestra el lugar donde se registró el extravío.

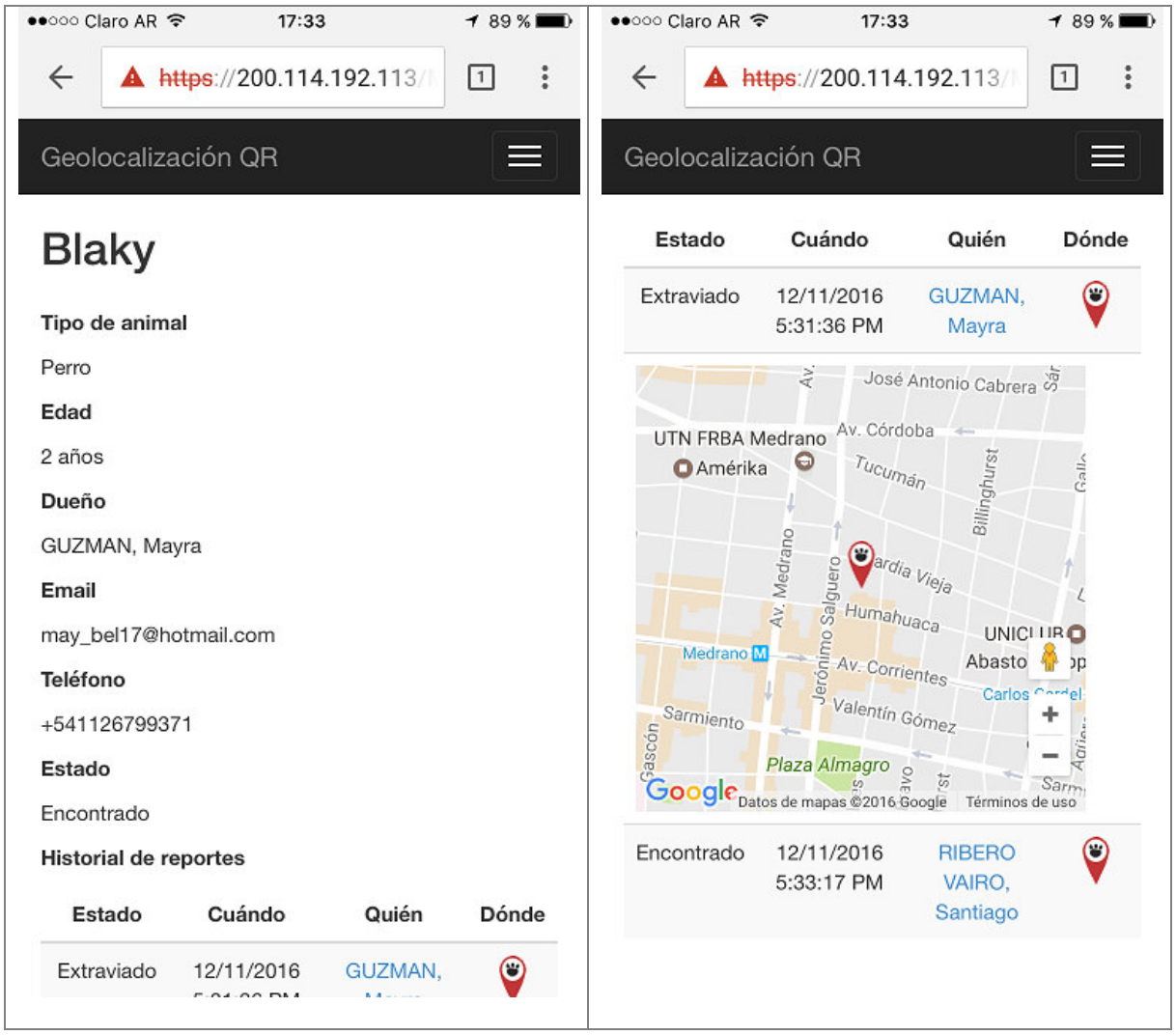


Figura 5.12: Vista de la ficha de la mascota Blaky por parte de un desconocido luego de haber leído su identificación.

Al mismo tiempo, por el hecho de que un usuario desconocido leyó el QR de la mascota extraviada, se le envía una notificación por SMS al dueño para avisarle de la aparición de su mascota. El aviso recibido se puede apreciar en la Figura 5.13. En este caso, el dueño recibe un aviso de que una persona encontró a su mascota y un teléfono para comunicarse, así puede coordinar el reencuentro con la misma.

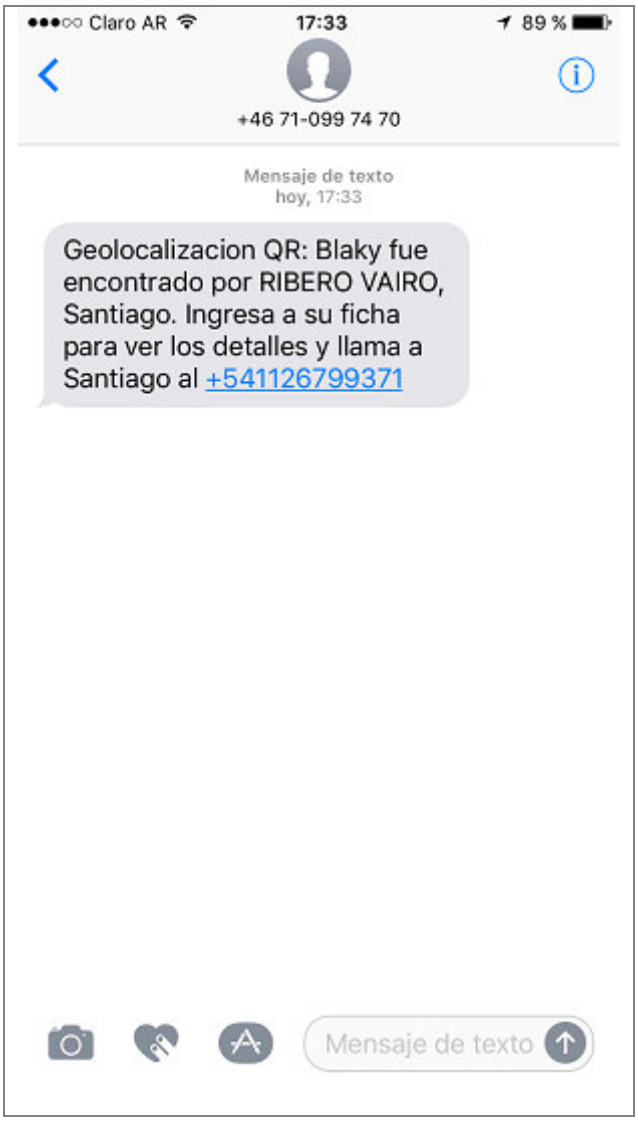


Figura 5.13: Notificación recibida por el dueño de Blaky en el momento en que un desconocido leyó su identificación.

Cuando el dueño de la mascota vuelve a ingresar a la ficha, encuentra el nuevo estado, indicando que la mascota fue encontrada por un desconocido y permitiéndole (al clic en el ícono correspondiente) ver en el mapa la ubicación en donde apareció, y (al clic en el nombre del usuario) ver la información de contacto de quien encontró a su mascota. Esto puede apreciarse en la Figura 5.14.

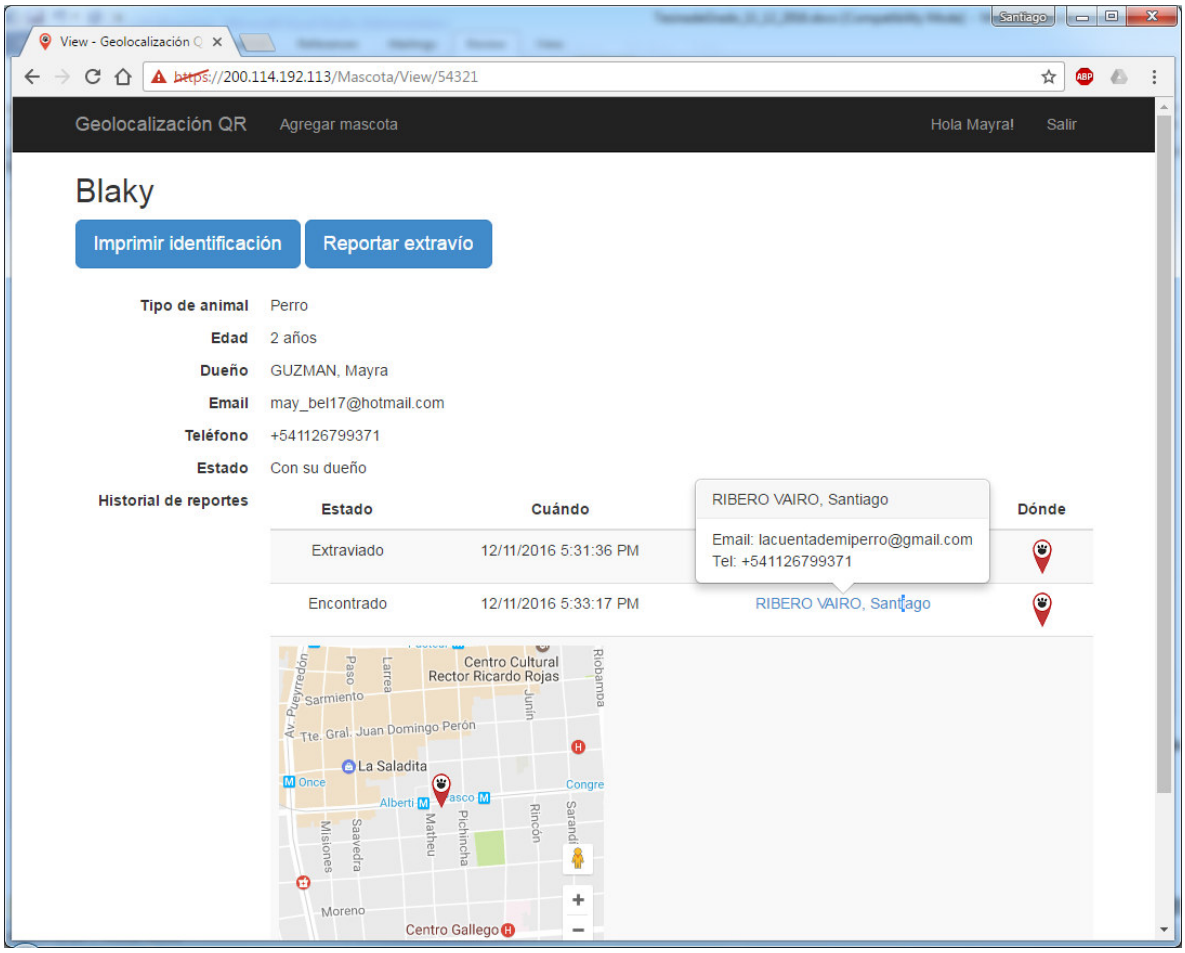


Figura 5.14: Vista de la ficha de la mascota Blaky por parte de su dueño luego de que fue encontrada por un desconocido.

Una vez que el dueño logra reunirse con la mascota, lee el código QR y se le envía al último usuario que encontró a la mascota una notificación de agradecimiento, indicándole la recompensa en caso de que el dueño haya definido una al reportar la mascota como extraviada. Esto se puede apreciar en la Figura 5.15.

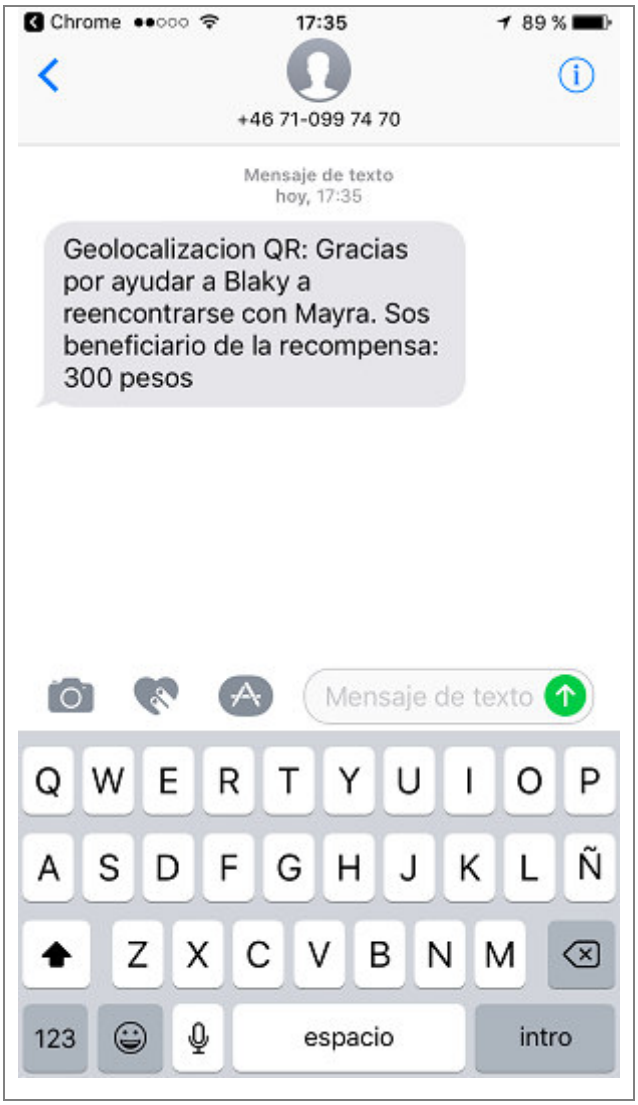


Figura 5.15: Notificación recibida por quien encontró a Blaky en el momento en que su dueño (por haberla recuperado) leyó su identificación.

Por otra parte, al haber leído el código QR, al dueño se le presenta la ficha de su mascota con el historial de estados actualizado. Esto se puede apreciar en la Figura 5.16.

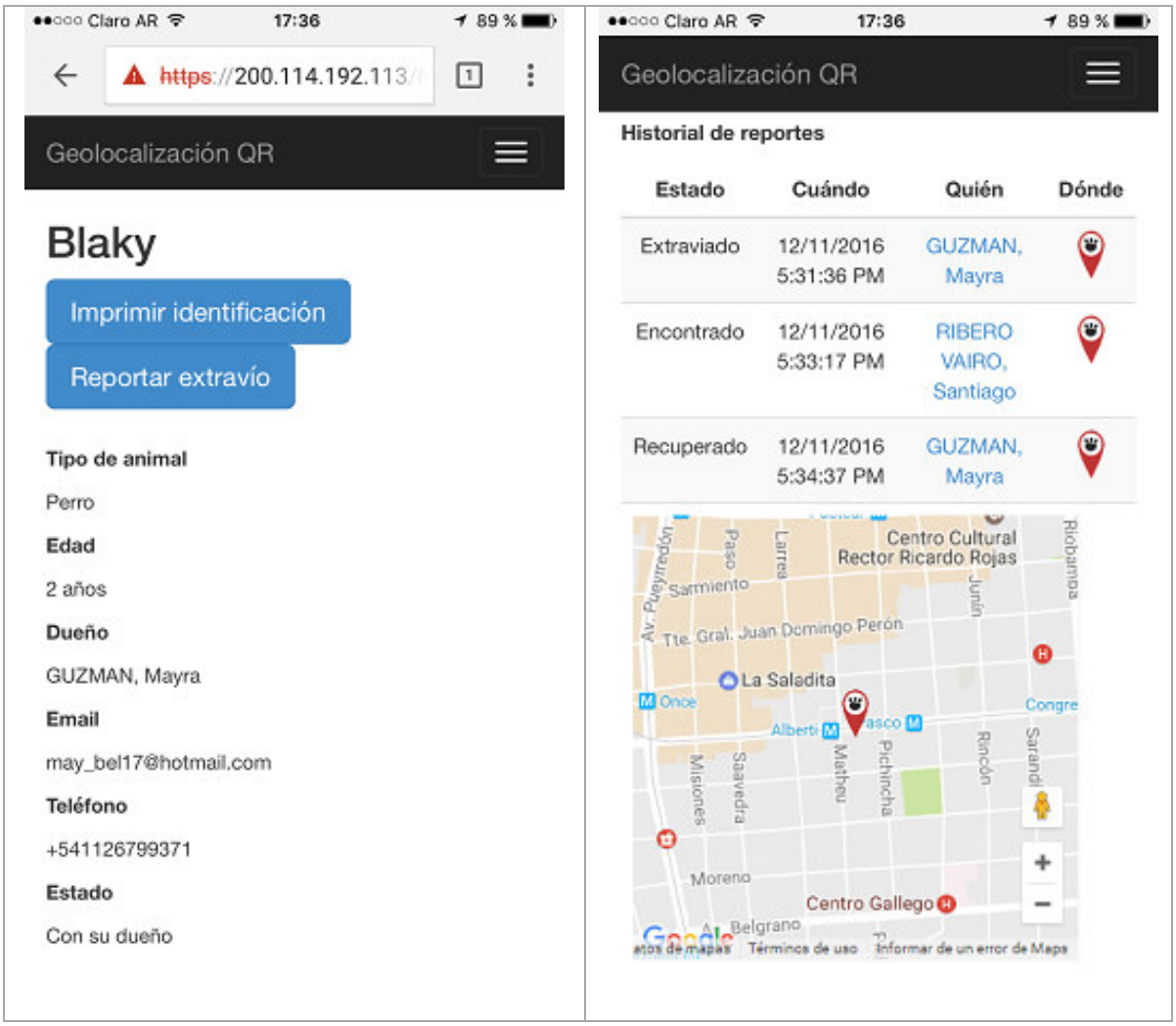


Figura 5.16: Ficha de Blaky vista por su dueño tras recuperarla.

Desde este momento, la mascota deja de presentarse en la lista de mascotas extraviadas de la *home page*.

Por último, todo usuario registrado cuenta con la posibilidad de incorporar mascotas propias al sistema. Esto puede hacerlo mediante el botón "*Agregar mascota*" de la cabecera, y completando el formulario, tal como se puede apreciar en la Figura 5.17.

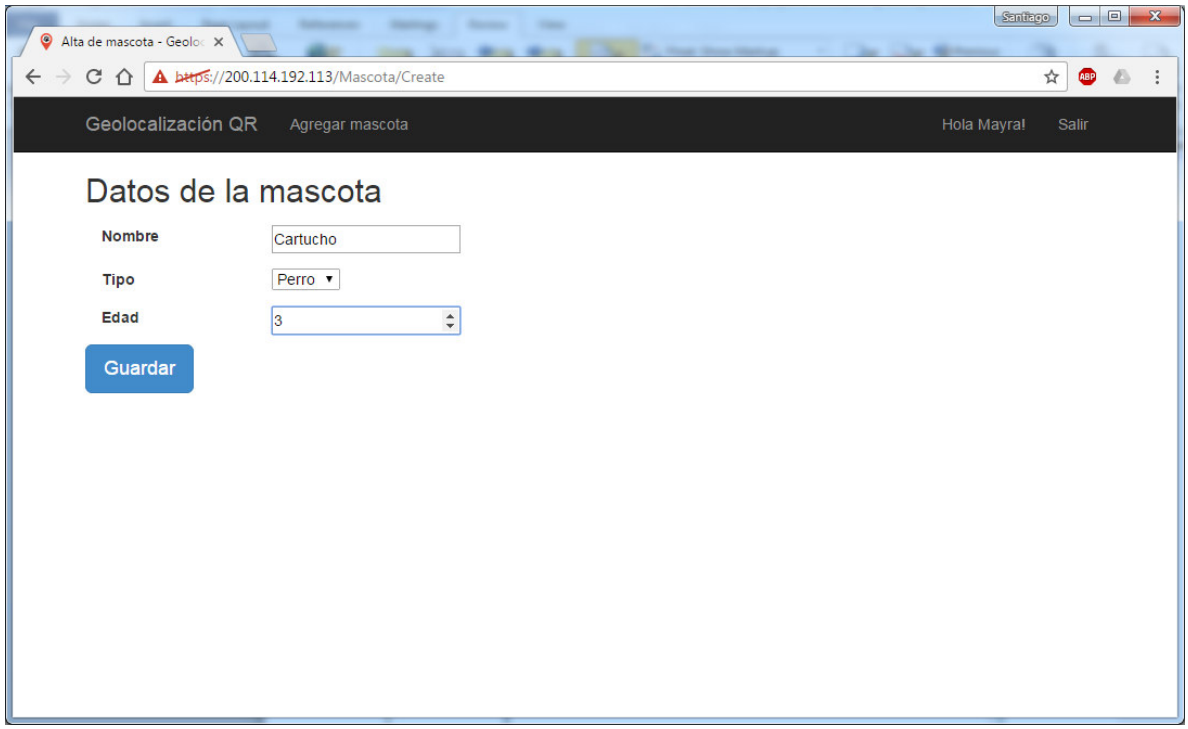


Figura 5.17: Formulario de alta de mascota.

Cabe aclarar que todas las posiciones tomadas de cada usuario se obtuvieron a partir de la API de posicionamiento de HTML5. Esto es transparente para el usuario.

De esta manera se pudo apreciar en este capítulo cómo el usuario puede interactuar con el mecanismo de lectura de QR como así también la integración la API de posicionamiento de HTML5 que permite tomar la posición de los usuarios, a medida que van generando distintos eventos relacionados con las mascotas.

Capítulo 6. Conclusiones y Trabajos Futuros.

En esta tesina se propuso una solución de modelado que contempla el posicionamiento de las mascotas (determinada por la ubicación de dueños y extraños al momento de identificar al animal), como así también sus estados de extravío que, complementados con un mecanismo de notificaciones automáticas, permiten la reubicación de mascotas extraviadas. Se pudieron apreciar en el Capítulo 3 las clases involucradas en el modelo propuesto como así también el funcionamiento del mismo mediante los distintos diagramas de secuencia.

Si bien el modelo propuesto soporta las notificaciones planteadas como parte de los objetivos (*Notificación del extravío de la mascota*, *Notificación del hallazgo de la mascota y Notificación de la recuperación de la mascota*), por como está planteado, dicho modelo puede ser extendido para considerar otras notificaciones.

En base al modelo propuesto se presentó un prototipo funcional. Esto nos permitió mostrar cómo se puede poner en práctica el modelo propuesto, en particular las notificaciones previamente mencionadas. Se pudo apreciar en el Capítulo 5 que las mascotas se identifican utilizando códigos QR mientras que la posición de los usuarios (ya sea del dueño o de la persona que lo encontró) se determina mediante la API de posicionamiento de HTML5.

Respecto del prototipo funcional, en el Capítulo 4 se describen algunos detalles técnicos que hubo que solucionar, y como los mismos fueron abordados.

Otro de los objetivos planteados en el Capítulo 1 era incorporar en el modelo la representación de *Necesidades* de las mascotas y *Recursos* de los usuarios, con el fin de aprovechar la extensibilidad del modelo para complementarlo con una funcionalidad que le de valor agregado. Al haberse indicado que esto formaría parte de los trabajos futuros, se detallará como último punto de este capítulo. De esta manera, se podrá observar una posible extensión específica que puede realizarse al modelo propuesto.

A continuación se mencionan posibles extensiones de esta tesina, tanto a nivel de diseño como para el prototipo. Estas extensiones fueron identificadas durante el desarrollo de la presente tesina.

- *Incorporación de fotografías*

  Se podría considerar, tanto en el diseño como en el prototipo, una funcionalidad que permita incorporar fotografías de cada mascota, lo que permitiría a los usuarios ver una secuencia de fotografías de animales recientemente extraviados, que al ser filtrados por tipo (por ejemplo "*perros*") en la zona en la que se encuentra, facilitará la detección de fichas de mascotas encontradas que (por el motivo que fuere) perdieron su código bidimensional. Los usuarios pueden buscar la ficha de la mascota en el sistema a través de las fotos y, al encontrarla, contactar al dueño.

- *Sistema como único medio de contacto entre usuarios*

  El sistema del modelo podría ser el único encargado de toda interacción entre usuarios, evitando así difundir información de contacto de los mismos. Es decir, que los usuarios (tanto dueños como personas que encuentras mascotas) interactuasen a través del sistema. A nivel del prototipo, este tendría la funcionalidad para, por ejemplo, permitir que los usuarios se llamen o se manden mensajes, sin que estos conozcan el número real del celular.

- *Selección de mecanismos de notificación*

  Se podría incorporar una funcionalidad que le permita al usuario seleccionar sus mecanismos de notificación, entre una lista de opciones.
  Si bien en el modelo propuesto los notificadores ya fueron modelados como una jerarquía extensible, tal como se muestra en la Figura 6.1, solo se especificó un notificador (NotificadorSMS). En caso de contarse con más notificadores, por ejemplo *NotificadorMail, NotificadorFacebook, NotificadorTwitter*, estos pasarían a ser subclases de *Notificador* y el usuario podría seleccionar sus notificadores. Esto en el prototipo, se implementaría permitiendo al usuario seleccionar el o los notificadores que desea que el sistema utilice para contactarlo.

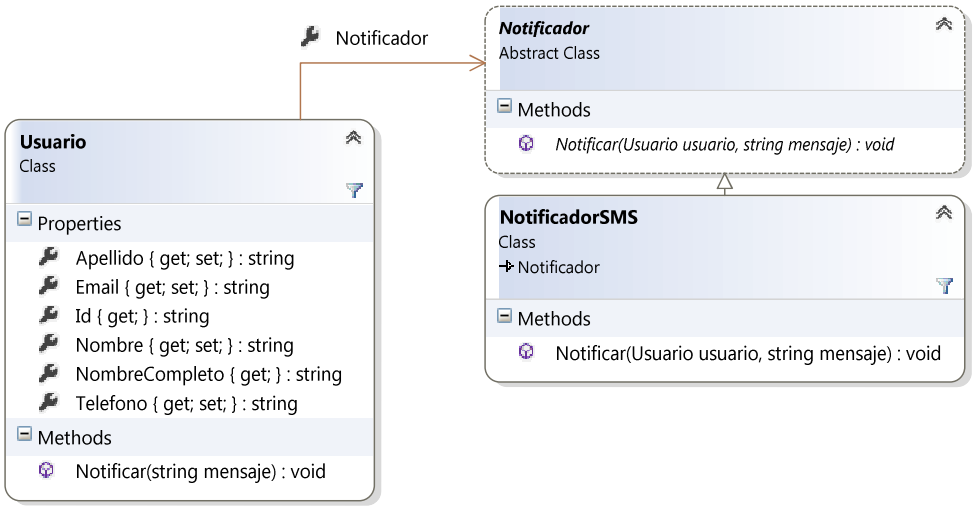


Figura 6.1: Jerarquía de notificadores del modelo propuesto.[45]

- *Desarrollar persistencia*

  El prototipo desarrollado como parte de la presente tesina no cuenta con un mecanismo de persistencia sino que al reiniciar el sistema, el mismo vuelve a inicializarse con instancias de prueba predefinidas. Se tomó esta decisión para facilitar el *deploy* del prototipo, dado que el foco estaba en probar la funcionalidad del modelo, la persistencia en una base de datos no era

[45] Se repite este diagrama del Capítulo 3 a fin de facilitar la comprensión de dónde se debería realizar dicha extensión.

crítica en este caso y por cuestiones de tiempo se decidió no incorporarla. En caso de querer implementar la persistencia, se recomienda construir una capa de "*repositorios*" que puedan ser utilizados por las entidades del dominio para persistirse, recuperarse, eliminarse o modificarse. Esto también llevaría a explorar las bases de datos para dispositivos móviles y cómo estas se deberían tener sincronizadas cada vez que hay información nueva de todas las mascotas del sistema. Esto sucedería en el caso de querer contar con una aplicación híbrida que pueda funcionar de manera off-line, es decir que el usuario podría estar por momentos sin conectividad, y para estos casos se podría usar los datos almacenados localmente en el dispositivo.

- *Evaluación de proximidad de mascota con posición de dueño*

  El prototipo no evalúa ninguna relación entre la posición de lectura de una mascota, y la posición de su *Dueño*. Se considera conveniente implementar esta evaluación como parte del procesamiento de la lectura de un código QR, de manera que la lectura de un desconocido produzca alertas y cambios de estado solo si la mascota se encuentra a una distancia considerable de su dueño. Esta situación podría suceder, por ejemplo, con mascotas que están acostumbradas a andar libres por la calle, entonces cada vez que un desconocido lee su código QR solo se dispara la alerta de extravío si la mascota está, supongamos, a más de 10 cuadras de donde vive el dueño (este radio de circulación normal de la mascota es definido por cada dueño). El dueño podría ser notificado de que alguien leyó el código QR pero solo como un aviso y además, por ejemplo, enviarle un agradecimiento al desconocido informándole que dicha mascota está dentro del área de la casa de su dueño.
  El modelo, tal como fue propuesto, cuenta con los componentes necesarios para incorporar esta evaluación, logrando de esta manera una extensión del mismo.

- *Necesidades y Recursos*

  Como se mencionó en el Capítulo 1 de la presente tesina, se buscó que el modelo propuesto contara con extensibilidad suficiente como para incorporar al dominio funcionalidades que potencien la utilidad del sistema. Una funcionalidad que considero de gran utilidad (aunque de menor interés técnico respecto al mecanismo de recuperación de mascotas estudiado) es la vinculación entre *Necesidades* de mascotas y *Recursos* de usuarios.
  Para lograr esta funcionalidad se propone incorporar el concepto de recurso y tipo de recurso como se detalla en la Figura 6.2. Se puede apreciar que las mascotas tienen necesidades y los usuarios tienen recursos. El sistema podría usar esta información para lograr que los usuarios puedan compartir sus recursos con las mascotas que los necesitan.

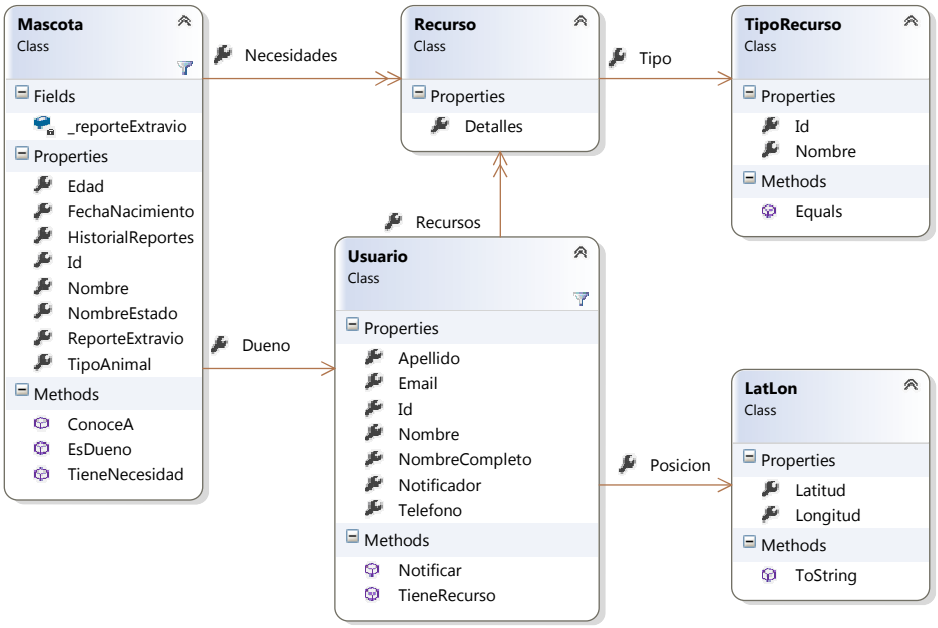


Figura 6.2: Incorporación de clases *Recurso* y *TipoRecurso*, relaciones y métodos para lograr la funcionalidad de *Necesidades* y *Recursos*.

Como se pudo apreciar en la Figura 6.2 la posición del *Usuario* será usada para lograr que tanto el dueño de la mascota como el que posee los recursos se puedan encontrar. Como se mencionó anteriormente, también se incorpora en *Mascota* y en *Usuario* relaciones a la clase Recurso (llamadas "*Necesidades*" y "*Recursos*", respectivamente). Dicha clase cuenta con una referencia a la nueva clase *TipoRecurso*, que representa las categorías de recursos, por ejemplo estas podrían ser:

- *Recursos materiales*
- *Económicos*
- *Profesionales*
- *Logísticos*
- *Alimenticios*
- *Medicamentos*
- *Familia adoptiva*
- *Voluntariado*

Por otra parte, la clase *Recurso* cuenta con una propiedad "*Detalles*" en la cual el usuario puede ingresar las particularidades y características del recurso puntual del que dispone o su mascota requiere. Por último, en la clase *Usuario* se incorporó el método "*TieneRecurso*" mientras que en *Mascota* "*TieneNecesidad*"; métodos que permiten conocer si existe relación entre esas entidades y una determinada categoría de recursos. Estos métodos son los que le permitirán a los usuarios filtrar por *TipoRecurso* para identificar usuarios que cuenten con determinado recurso o mascotas que lo requieran. Esta funcionalidad también se podría llevar al prototipo logrando así la extensión del mismo con nueva funcionalidad.

Otra gran ventaja de hacer estas incorporaciones al modelo es que representan una forma de determinar las necesidades que puede tener una mascota al momento de hallarla extraviada. Por ejemplo, si un desconocido encuentra a una mascota podría no lograr contactar al dueño, pero podría identificar (en la propia ficha de la mascota) que la misma requiere una medicación particular. O podría identificar que la mascota busca una familia adoptiva, teniendo la posibilidad de adoptarla él mismo.

De esta manera quedan detalladas algunas extensiones concretas del modelo propuesto y como se podrían llevar a cabo.

Referencias bibliográficas.

[Biao, 2007] Biao, L. (2007, August). A DataMatrix-based mutant code design and recognition method research. In Image and Graphics, 2007. ICIG 2007. Fourth International Conference on (pp. 570-574). IEEE.

[Borkowski, 1994] Borkowski, D., Das, P., & Tao, F. C. (1994, October). Selection of a processor for a Portable MaxiCode Reader. In Digital Signal Processing Workshop, 1994., 1994 Sixth IEEE (pp. 7-10). IEEE.

[Bui, 2014] Robust Message Hiding for QR Code. Thach V. Bui∗, Nguyen K. Vu∗, Thong T.P. Nguyen∗, Isao Echizen† and Thuc D. Nguyen∗

[Gamma et. al, 2003] E. Gamma, R. Helm, R. Johnson y J. Vlissides, Patrones de diseño. Elementos de software orientado a objetos reutilizable, Madrid: Pearson Educación, 2003.

[Hersh et. Al. 2008] Hersh, M., & Johnson, M. A. (Eds.). (2010). Assistive technology for visually impaired and blind people. Springer Science & Business Media.

[Honkova, 2013] Honkova, J. (2013). The Russian Federation's Approach to Military Space and Its Military Space Capabilities. George C. Marshall Institute.

[ISO/IEC 15438:2015, 2015] ISO/IEC 15438:2015 - Information technology -- Automatic identification and data capture techniques -- PDF417 bar code symbology specification.

[ISO/IEC 16022:2006, 2006] ISO/IEC 16022:2006 Information technology -- Automatic identification and data capture techniques -- Data Matrix bar code symbology specification.

[ISO/IEC 16023:2000, 2000] ISO/IEC 16023:2000 - Information technology -- International symbology specification – MaxiCode.

[ISO/IEC 18004:2000, 2000] ISO/IEC 18004:2000 - Information technology -- Automatic identification and data capture techniques -- Bar code symbology -- QR Code.

[ISO/IEC 24778:2008, 2008] Information technology -- Automatic identification and data capture techniques -- Aztec Code bar code symbology specification.

[Kato & Tan, 2007] Kato, H. & Tan, K.T. (2007). Pervasive 2D Barcodes for Camera Phone Applications, Pervasive Computing, IEEE , Vol.6, No.4, (October, 2007), pp.76-85.

[Magro, 2014] R. Magro. Graphics codes and applied entomology dynamic processing. Boletín de la SEA, 54, 30.

[Parikh, 2008] Parikh, D., & Jancke, G. (2008, January). Localization and segmentation of a 2D high capacity color barcode. In Applications of Computer Vision, 2008. WACV 2008. IEEE Workshop on (pp. 1-6). IEEE.

[Punetha, 2014] Punetha, D., & Mehta, V. (2014, September). Protection of the child/elderly/disabled/pet by smart and intelligent GSM and GPS based automatic tracking and alert system. In Advances in Computing, Communications and Informatics (ICACCI, 2014 International Conference on (pp. 2349-2354). IEEE.

[Subpratatsavee, 2014] Subpratatsavee, P., & Kuacharoen, P. (2014, November). An Implementation of a Paper Based Authentication Using HC2D Barcode and Digital Signature. In IFIP International Conference on Computer Information Systems and Industrial Management (pp. 592-601). Springer Berlin Heidelberg.

[Xingxing Li et. al, 2014] Li, X., Ge, M., Dai, X., Ren, X., Fritsche, M., Wickert, J., & Schuh, H. (2014). Accuracy and reliability of multi-GNSS real-time precise positioning: GPS, GLONASS, BeiDou, and Galileo. Journal of Geodesy, 89(6), 607-635.

[Yonezawa, 2009] Yonezawa, K., Miyaki, T., & Rekimoto, J. (2009, October). Cat@ Log: sensing device attachable to pet cats for supporting human-pet interaction. In Proceedings of the International Conference on Advances in Computer Enterntainment Technology (pp. 149-156). ACM.

[Zandbergen, 2009] Zandbergen, P. A. (2009). Accuracy of iPhone locations: A comparison of assisted GPS, WiFi and cellular positioning. Transactions in GIS, 13(s1), 5-25.

[Zhang, 2008] Zhang, P., Dai, S., & Mu, P. A. (2008). Research on information recognition method of image for PDF417 two-dimension bar code. In 2008 7th World Congress on Intelligent Control and Automation.

[Zhang, 2010] Ke, H., & Zhang, G. (2010). An algorithm correcting Flex distortion of Aztec code. In 2010 2nd IEEE International Conference on Information Management and Engineering.