Proyecto de Análisis y Diseño de Software: Entrega 2

Ana Calzada, Leandro García y Fabián Gutiérrez (*VecindApp*)

2 de marzo de 2020

1. Diagrama de clases

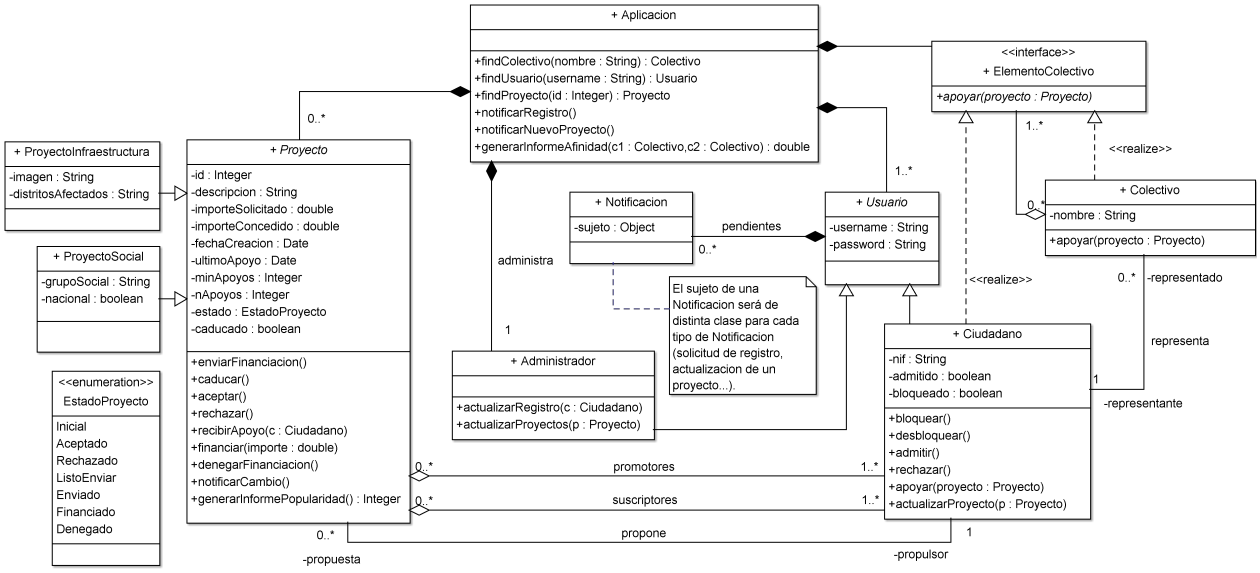

Figura 1: Diagrama de clases del proyecto.

El diagrama de la figura 1 consta de cuatro clases principales: `Aplicacion`, *`Proyecto`*, *`Usuario`* y `Colectivo`.

- La clase `Aplicacion` funciona como un contenedor del conjunto de objetos del resto de clases que maneja la aplicación. Por consiguiente, sus métodos permiten buscar objetos (y comprobar así si existen en el sistema), notificar al administrador sobre nuevos registros de ciudadanos y de proyectos, y comparar dos colectivos.

- La clase *`Proyecto`* (de la que heredan `ProyectoInfraestructura` y `ProyectoSocial`) almacena la información relevante de un proyecto, además de su estado dentro de su ciclo de vida. Además, almacena un conjunto de ciudadanos promotores (esto es, que apoyan el proyecto) y uno de suscriptores (a quienes notifica sus cambios de estado).

- La clase *`Usuario`* engloba las subclases `Administrador` y `Ciudadano`. Un usuario tiene un conjunto de objetos `Notificacion` pendientes, cada uno asociado a un objeto que determina el tipo de notificación (solicitud de registro de un ciudadano, cambio de estado de un

proyecto...). La clase `Administrador` tiene métodos que actualizan la lista de notifiaciones pendientes, bien sea una nueva solicitud de registro o una nueva propuesta de proyecto.

La clase `Ciudadano` hereda también de la interfaz *`ElementoColectivo`* y, por tanto, implementa el método `apoyar`, lo que permite a un ciudadano apoyar individualmente un proyecto. Además, de manera similar a `Administrador`, tiene un método `actualizarProyecto` que añade a su lista de pendientes una notificación asociada a un proyecto (al que está suscrito) que ha cambiado de estado.

- Finalmente, la clase `Colectivo` tiene un conjunto de `ElementoColectivo` y, a su vez, hereda de dicha interfaz. Asimismo, implementa el método `apoyar`, lo que permite apoyar colectivamente un proyecto.

Cabe destacar que el diagrama no incluye métodos constructores, *setters*, *getters* o *adders*, entre otros, que sean triviales.

2. Diagramas de transición de estados

2.1. Clase `Ciudadano`

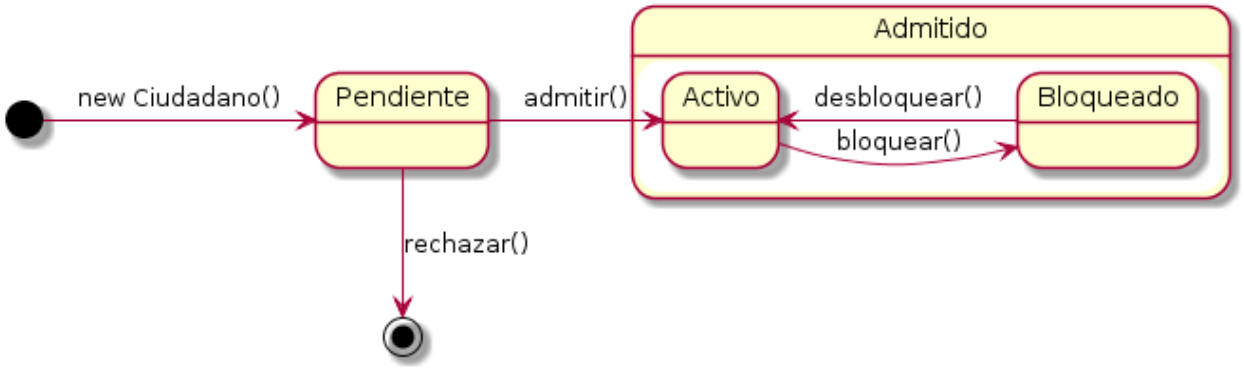

Figura 2: Diagrama de transición de estados de la clase `Ciudadano`

Este diagrama muestra la evolución de un elemento de la clase `Ciudadano`. Una vez el objeto se crea (a partir de la solicitud de un ciudadano aún no registrado) pasa al estado `Pendiente`, en espera de ser aceptado o rechazado por el administrador.

Una vez el usuario ha sido aceptado y se encuentra dentro del estado jerárquico `Admitido`, este pasará a estar activo inicialmente, pudiendo ser bloqueado y desbloqueado por el administrador en cualquier momento.

En caso de que el administrador rechace el registro, el objeto `Ciudadano` será eliminado.

2.2. Clase `Proyecto`

En este diagrama también se comienza con un estado `Propuesto`, en el cual se espera a que el administrador decida aceptarlo o rechazarlo, al igual que en el caso de `Ciudadano`.

En caso de ser aceptado, el proyecto entra en un estado `En Votación`, formado por dos componentes ortogonales. Inicialmente, el objeto se encontrará en los estados `No Listo` y `No Caducado`, respectivamente. Mientras se encuentre en este último estado, el proyecto podrá seguir recibiendo

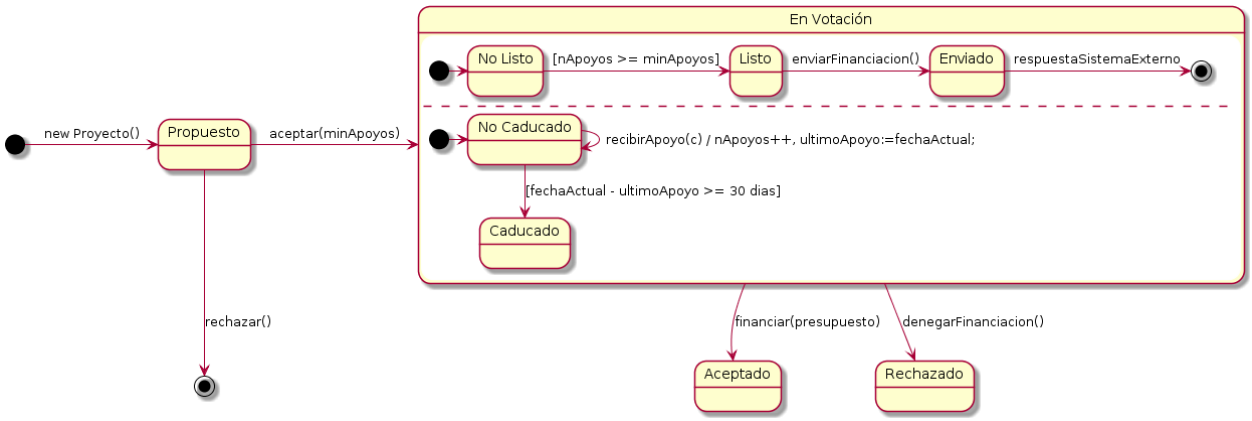

Figura 3: Diagrama de transición de estados de la clase `Proyecto`

votos. En el caso de que transcurran 30 días sin que el proyeecto sea votado, este pasará al estado de `Caducado`, donde ya no podrá ser votado. En cuanto al otro componente, el estado pasará de `No Listo` a `Listo` cuando se alcance un número de apoyos estabecido previamente por el administrador. Una vez se encuentre en `Listo`, podrá ser enviado para financiación por su propietario, y pasará, por consiguiente, al estado `Enviado`.

En el momento que el objeto `Proyecto` recibe una respuesta por parte del sistema externo, este sale del estado `En Votación`, bien para pasar a `Rechazado` (si el sistema externo desestima la propuesta) o a `Aceptado` (en caso de que se le conceda un presupuesto).

3. Diagramas de secuencia

3.1. Apoyar un proyecto (individualmente)

El diagrama de la figura 4 describe las interacciones asociadas a apoyar individualmente un proyecto. Por simplicidad, se asume como precondición que el ciudadano ha hecho login con éxito (la aplicación apunta al objeto `c:Ciudadano` asociado al usuario actual).

Cabe destacar que la secuencia para apoyar colectivamente un proyecto es similar a esta, añadiendo que el ciudadano ha de indicar en nombre de qué colectivo va a apoyar el proyecto y que la implementación de `apoyar` de `Colectivo` consiste en llamadas al método `apoyar` de `Ciudadano` por cada ciudadano perteneciente a dicho colectivo.

3.2. Enviar un proyecto a financiación

El diagrama de la figura 5 describe las interacciones que se dan a la hora de que un ciudadano envíe su proyecto a financiación. Se presupone que el ciudadano ya hizo *login* y se encuentra ante la lista de proyectos que ha creado y han sido aceptados por el administrador (de hecho, estas son las precondiciones del caso de uso). También se asume que el sistema externo de financiación está activo a la espera de recibir propuestas a considerar, por ello está activa su línea de vida.

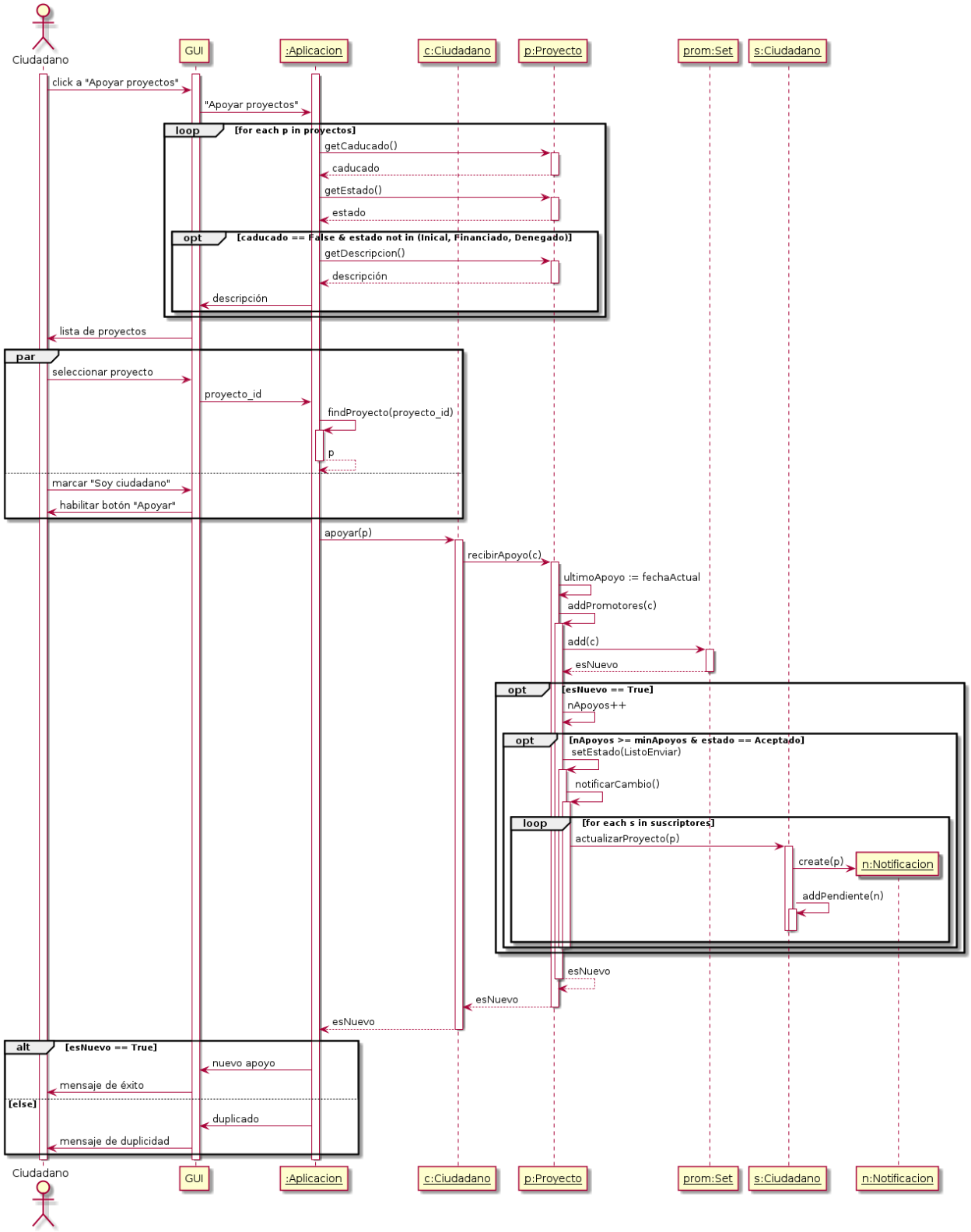

Figura 4: Diagrama de secuencia para el escenario de apoyar un proyecto individualmente.

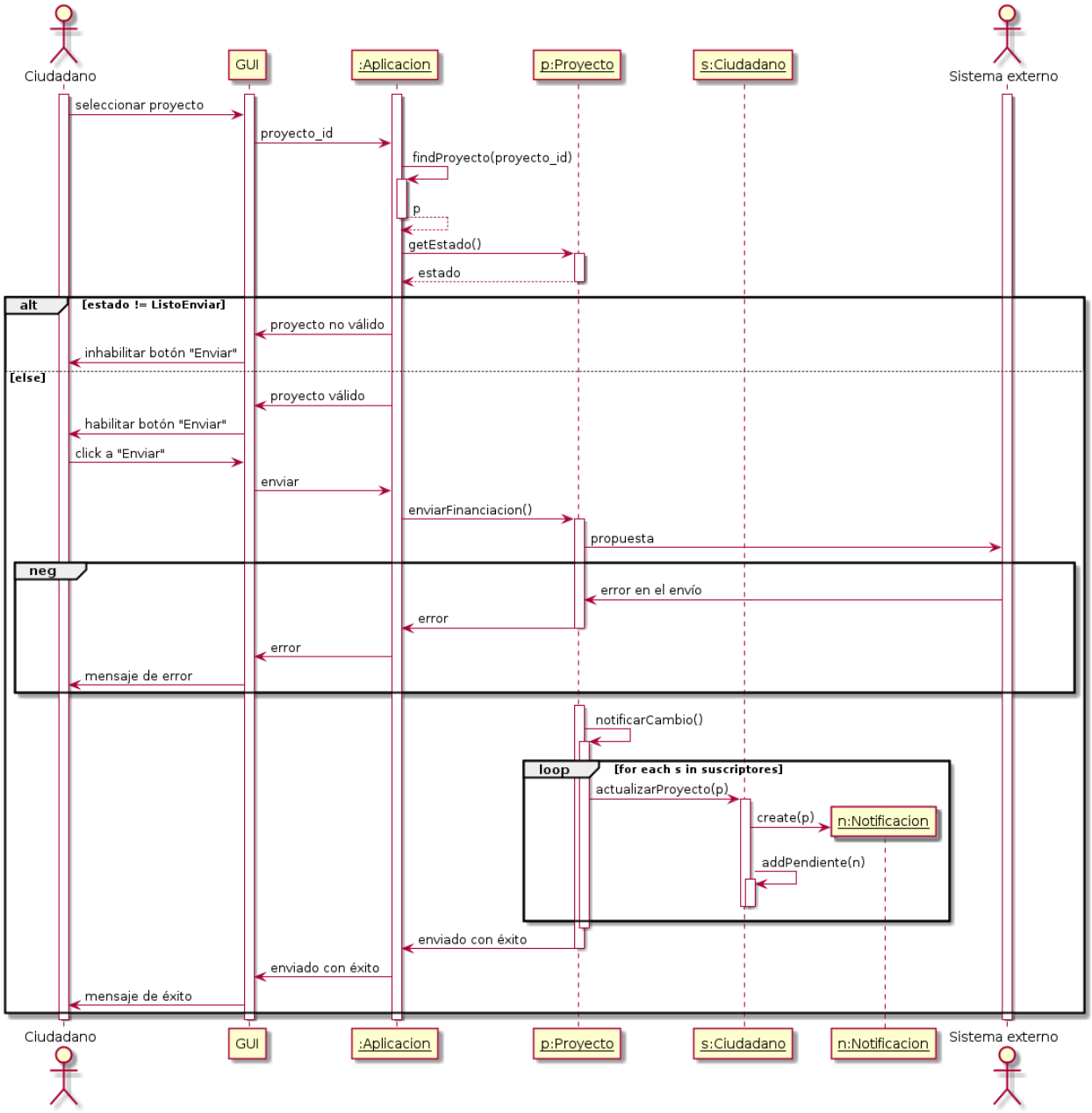

Figura 5: Diagrama de secuencia para el escenario de enviar un proyecto a financiación.