

ANEXO IV: DISEÑO DEL SISTEMA

Habitr - Aplicación para fomentar los hábitos saludables.

Trabajo de Fin de Grado

Ingeniería Informática



VNiVERSIDAD D SALAMANCA

Junio 2023

Autor:

Jorge Cruz García

Tutores:

André Filipe Sales Mendes

Gabriel Villarubia González

1 TABLA DE CONTENIDO

Índice de figuras	3
Índice de tablas.....	4
2 Introducción.....	5
3 Modelo de diseño	5
3.1 Patrones arquitectónicos	6
3.1.1 BLoC	6
3.1.2 Singleton	6
3.2 Clases de diseño	7
3.2.1 Capa Vista	7
3.2.2 Capa BLoC	9
3.2.3 Capa Repositorio	10
3.2.4 Capa Datos.....	10
3.3 Vista Arquitectónica	12
3.4 Realización de casos de uso.....	13
3.4.1 Diagrama del paquete gestión autenticación.....	13
3.4.2 Diagrama del paquete gestión usuarios.....	16
3.4.3 Diagrama del paquete gestión feed	17
3.4.4 Diagrama del paquete gestión social	20
3.4.5 Diagrama del paquete gestión estadísticas	22
3.4.6 Diagrama del paquete gestión rutinas	22
3.4.7 Diagrama del paquete gestión logros	25
3.4.8 Diagrama del paquete gestión notificaciones:	26
3.4.9 Diagrama del paquete gestión juego	27
4 Diseño de base de datos.....	29
5 Modelo de despliegue	30
6 Referencias.....	31

ÍNDICE DE FIGURAS

Figura 1: Capa vista.....	8
Figura 2: Capa BLoC	9
Figura 3: Capa repository	10
Figura 4: Capa datos	11
Figura 5: Vista arquitectónica.....	12
Figura 6: Diagrama de secuencia registro	13
Figura 7: Diagrama de secuencia login.....	14
Figura 8: Diagrama de secuencia logout	15
Figura 9: Diagrama de secuencia recuperar contraseña	16
Figura 10: Diagrama de secuencia ver perfil	16
Figura 11: Diagrama de secuencia crear publicación.....	17
Figura 12: Diagrama de secuencia borrar publicación	18
Figura 13: Diagrama de secuencia ver feed.....	18
Figura 14: Diagrama de secuencia dar like	19
Figura 15: Diagrama de secuencia compartir publicación.....	19
Figura 16: Diagrama de secuencia enviar solicitud de amistad	20
Figura 17: Diagrama de secuencia compartir código de amigos	20
Figura 18: Diagrama de secuencia ver solicitudes de amistad	21
Figura 19: Diagrama de secuencia responder a petición de amistad	21
Figura 20: Diagrama de secuencia ver estadísticas	22
Figura 21: Diagrama de secuencia crear rutina.....	22
Figura 22: Diagrama de secuencia editar rutina	23
Figura 23: Diagrama de secuencia borrar rutina	24
Figura 24: Diagrama de secuencia ver rutinas públicas.....	24
Figura 25: Diagrama de secuencia añadir rutina pública	25
Figura 26: Diagrama de secuencia completar rutina	25
Figura 27: Diagrama de secuencia desbloquear logro	26
Figura 28: Diagrama de secuencia ver logros	26
Figura 29: Diagrama de secuencia programar notificaciones	27
Figura 30: Diagrama de secuencia ver juego	27
Figura 31: Diagrama de secuencia cambiar flores	28
Figura 32: Diseño de la base de datos	29
Figura 33: Diagrama de despliegue	30

ÍNDICE DE TABLAS

2 INTRODUCCIÓN

En este anexo, *Anexo IV – Diseño del sistema* se recoge toda la documentación a la fase de documentación referente a la fase de diseño del sistema. La fase de diseño se centra en el dominio de la solución directamente, y representa el sistema a su nivel más cercano a la interpretación.

Como está a un nivel más bajo, los nombres de clases, métodos y atributos son muy parecidos o iguales a los que se verán en la implementación de la aplicación.

El documento tiene la siguiente estructura:

- **Modelo de Diseño:** el modelo de diseño sigue los siguientes puntos:
 - **Patrones arquitectónicos:** Se explicarán los patrones arquitectónicos usados durante el diseño.
 - **Subsistemas de diseño:** Se hace una organización general del sistema en paquetes, y las relaciones que tienen entre ellos.
 - **Clases de diseño:** Se enseñan las clases de diseño, con sus atributos, operaciones y relaciones.
 - **Vista arquitectónica:** Se enseña una vista detallada de las capas del patrón BLoC que fue usado en la arquitectura.
 - **Realización de casos de uso:** Se muestran los diagramas de secuencia para casos de uso del sistema donde se detallan los intercambios de mensajes entre objetos y clases.
- **Diseño de la base de datos:** Es un diagrama entidad-relación explicando toda la información que tiene que manejar el sistema
- **Modelo de despliegue:** Muestra el despliegue de los nodos y artefactos del sistema.

3 MODELO DE DISEÑO

En la fase del desarrollo se ha decidido utilizar ciertos patrones arquitectónicos que facilitarán el desarrollo. Se utiliza el patrón **BLoC** a la hora de organizar la arquitectura general de la aplicación.

Se hará uso de las siguientes tecnologías:

- **Flutter** para el desarrollo de vistas
- **Flutter_bloc:** Modulo que implementa el patrón BLoC, explicado a continuación, para su uso fácil con Flutter.
- **Supabase:** Base de datos distribuida que gestiona el acceso a los datos, además de la autenticación.
- **Flutter_supabase:** Módulo de Flutter que implementa directamente las interacciones con Supabase, gestionando directamente las llamadas a SQL y las funciones de autenticación.

3.1 PATRONES ARQUITECTÓNICOS

3.1.1 BLoC

El patrón **BLoC**, o **Business Logic Component**, es un patrón utilizado para separar de forma directa la capa de vista de la capa de negocio y de datos del sistema. El patrón está formado por tres componentes:

- **Vista:** La **vista** se encarga de la representación visual de datos y su interactividad con el usuario; representa los datos del sistema a través de distintas pantallas y *widgets*. La asociación de vista a datos son muchas a uno, es decir, los datos contenidos en un sólo modelo son representados por distintas interfaces gráficas.
- **BLoC:** El **BLoC** es un componente que **encapsula la lógica de negocio de la aplicación**. Se distingue del controlador del patrón **MVC** en el hecho de que es controlado **por eventos** en vez de lanzando funciones directamente, y que **muestra los resultados a través de cambios entre ciertos estados definidos** (por ejemplo, estado “cargado”, estado “error”...).
- **Repositorio:** Un **repositorio** tiene como objetivo **encapsular la capa de datos y hacerse cargo de las interacciones con el sistema de bases de datos**. Es decir, el objetivo de este componente es **ser capaz de proveer, acceder y modificar los datos que sean necesarios para el funcionamiento de un BLoC**.

El patrón BLoC presenta varias ventajas:

- **Separación de responsabilidades** a la hora de desarrollar los distintos componentes. Un BLoC no tiene que saber **cómo se están obteniendo unos datos del repositorio**, y una vista **no tiene que entender cómo está funcionando la lógica interna del BLoC**. De esta forma, todos los componentes se pueden desarrollar de forma separada, y si surgen cambios necesarios de funcionamiento o infraestructura (por ejemplo, necesidad de implementar nuevas funcionalidades, o un cambio en el sistema de bases de datos utilizado) **sólo necesitaremos cambiar uno de los módulos, y el resto seguirán funcionando de la misma manera**.
- **Alta cohesión y bajo acoplamiento entre los componentes de las distintas capas**.
- **Uso de eventos y estados** que permiten de una forma mucho más simple **manejar el estado de la vista**.

3.1.2 Singleton

El patrón **Singleton**, o de instancia única, es un patrón de diseño que permite restringir la creación de ciertos objetos a **un único objeto global**.

Su intención es garantizar que una clase **sólo tenga una instancia, y que sea de acceso global**.

Su implementación consiste en definir un método estático dentro de la clase, que cree el objeto si no existe, y si el objeto ya existe, devolver la instancia ya creada.

Este patrón es visto cómo un *antipatrón* por algunos desarrolladores, puesto que introduce el **problema de las variables globales** en nuestras aplicaciones, y por eso es utilizado

dentro de la aplicación de forma conservadora, para clases en las cuales sólo necesitamos una instancia y en las cuales la instanciación de la clase es más costosa (por ejemplo, el gestor de notificaciones, o la instancia de conexión con la base de datos).

3.2 CLASES DE DISEÑO

En este apartado veremos los contenidos de cada capa del patrón **BLoC**, además de las clases de cada paquete y los métodos de dichas clases.

3.2.1 Capa Vista

En la siguiente figura se ve cómo en la capa vista hay un subpaquete por cada uno de los módulos previamente definidos de la aplicación web. Dentro de cada paquete están los archivos que definen cada una de las vistas. El paquete **Layout** define la pantalla principal de la aplicación, que coordinará el resto de pantallas.

Anexo IV: Diseño del Sistema

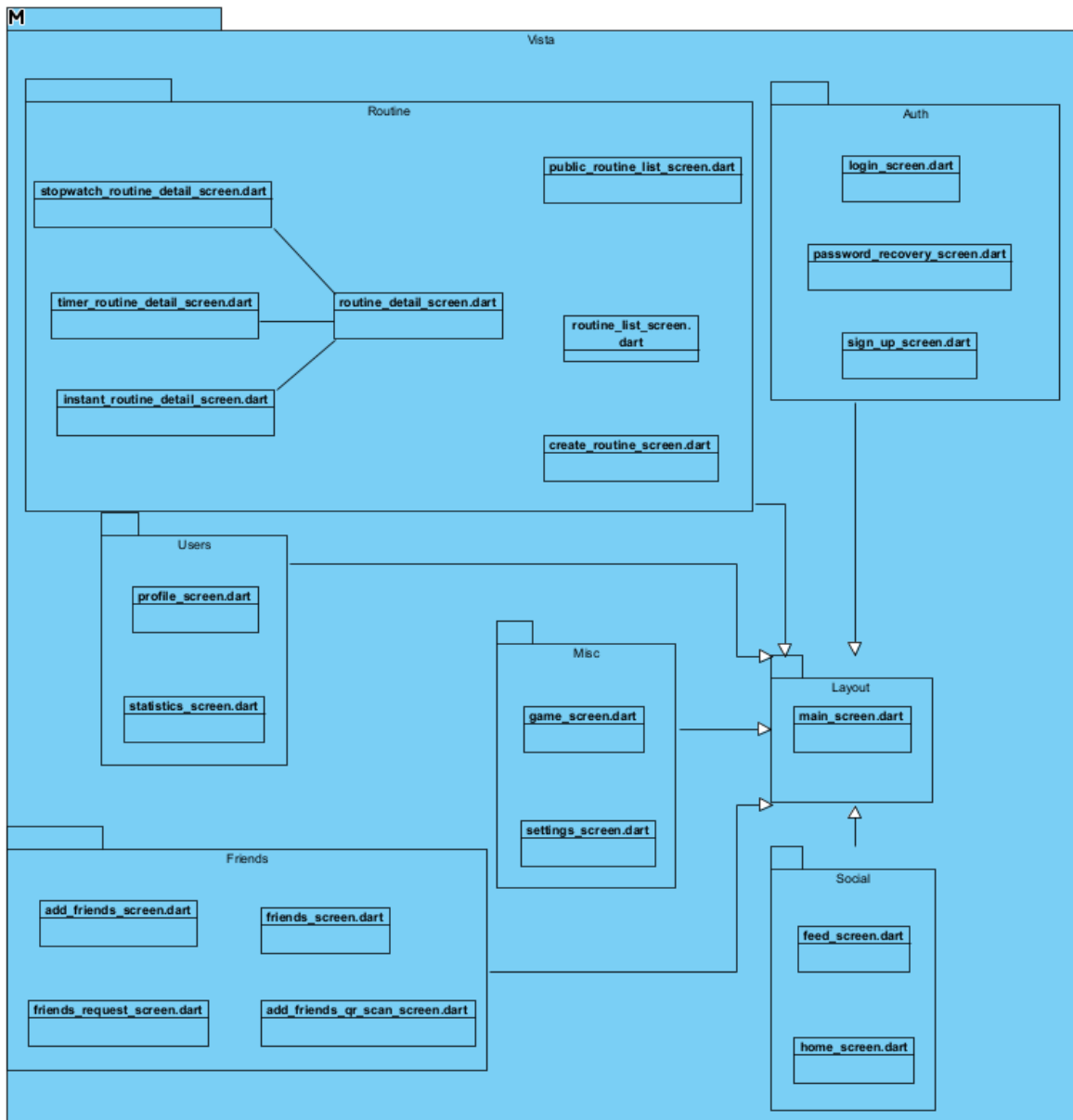


Figura 1: Capa vista

3.2.2 Capa BLoC

En la siguiente figura, se explica la capa BLoC, que actúa de manera parecida a lo que haría la capa controlador en el MVC. Todas las clases tienen métodos llamados *onNombreEvento*, que se lanzarán de forma automática cuando el **BLoC** reciba dicho evento. Además, incluimos aquí el singleton que gestionará las notificaciones, que a pesar de que no sea un BLoC por razones prácticas, es una clase que encapsula el control de cierta funcionalidad de la aplicación en un objeto fácilmente accesible a través del patrón singleton.

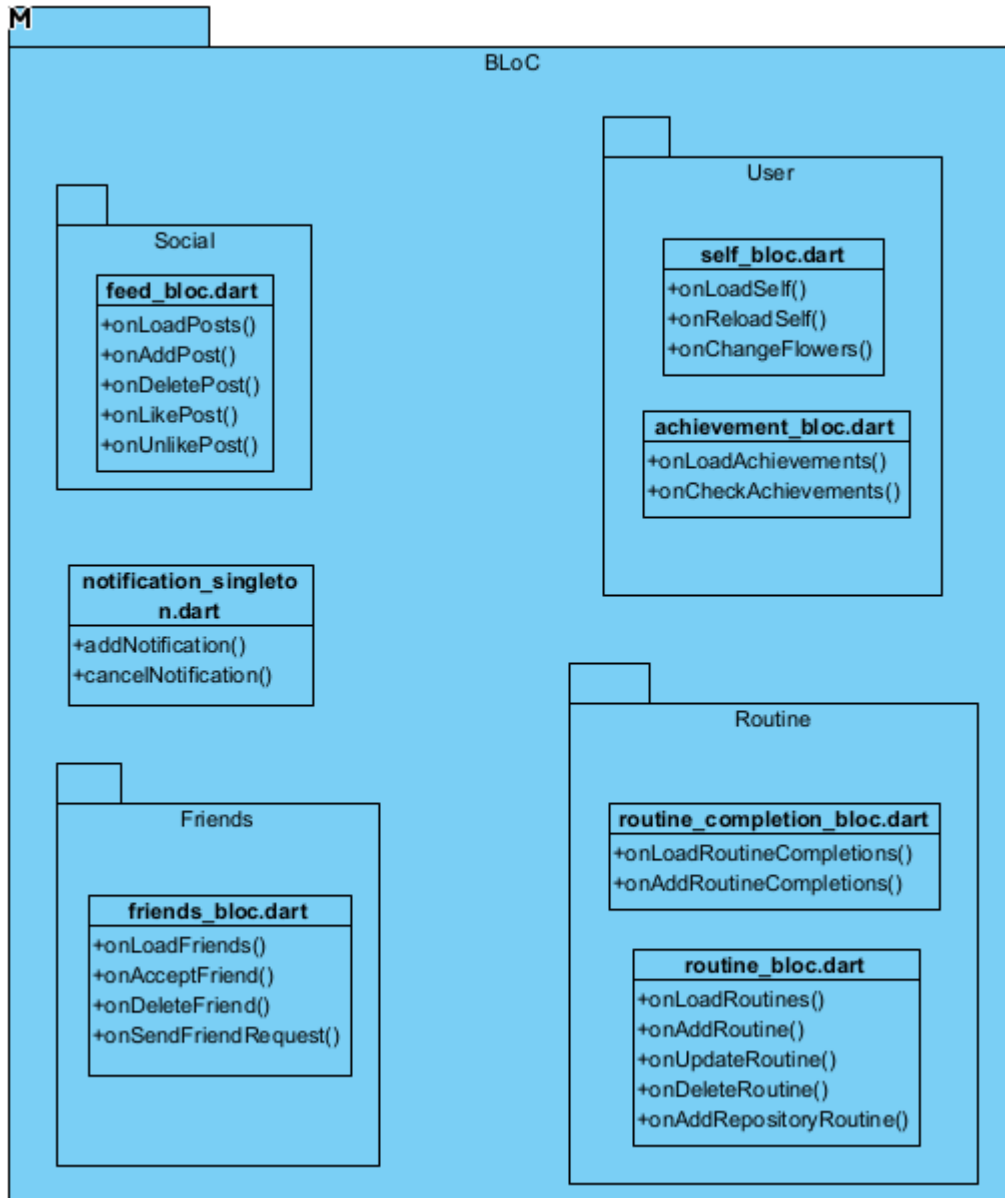


Figura 2: Capa BLoC

3.2.3 Capa Repositorio

En la siguiente figura, se explica la estructura de la capa Repositorio, que provee los datos a la **capa BLoC** abstrayendo el acceso a dichos datos.

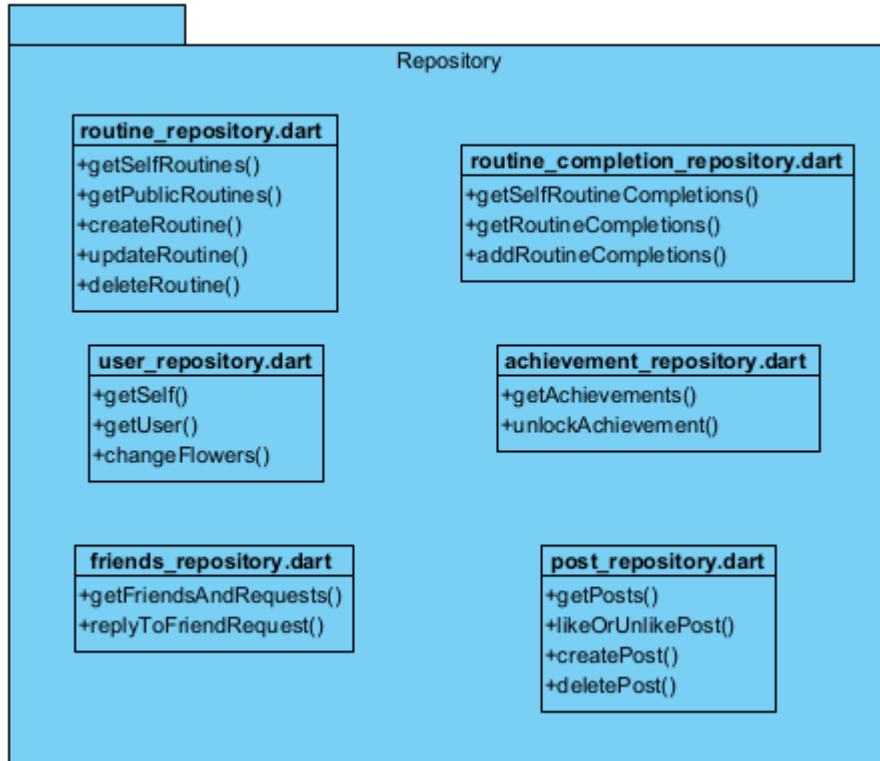


Figura 3: Capa repository

3.2.4 Capa Datos

La siguiente figura explica la capa de datos, que define las clases básicas de nuestra aplicación a un nivel cercano al de la base de datos. Son las clases básicas que instanciamos y representamos a lo largo de la aplicación.

3.3 VISTA ARQUITECTÓNICA

En el diagrama de la figura se puede ver representada de forma esquemática el esquema que sigue nuestra aplicación, haciendo uso del patrón BLoC.

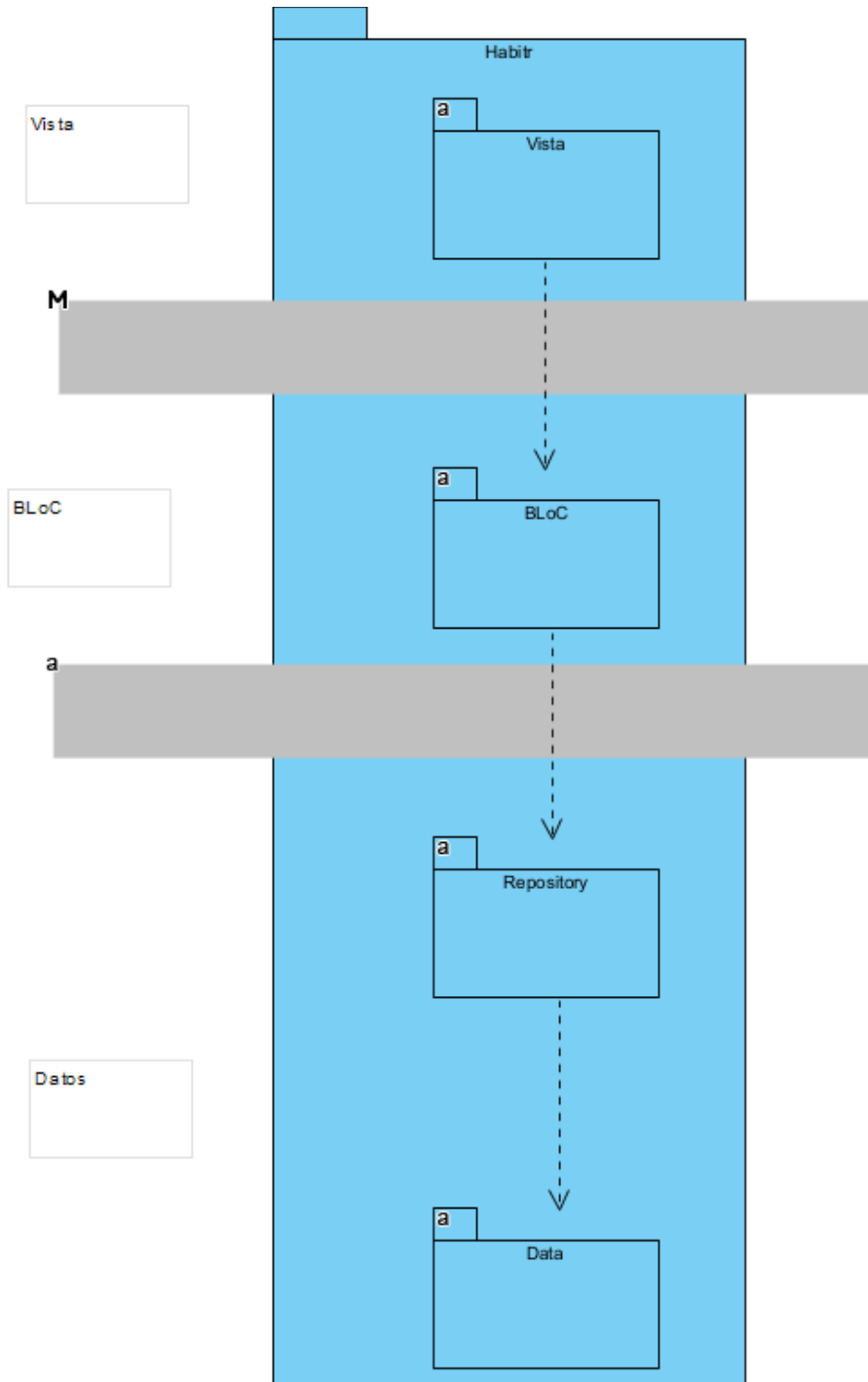


Figura 5: Vista arquitectónica

3.4 REALIZACIÓN DE CASOS DE USO

3.4.1 Diagrama del paquete gestión autenticación

Caso de uso **Registro**:

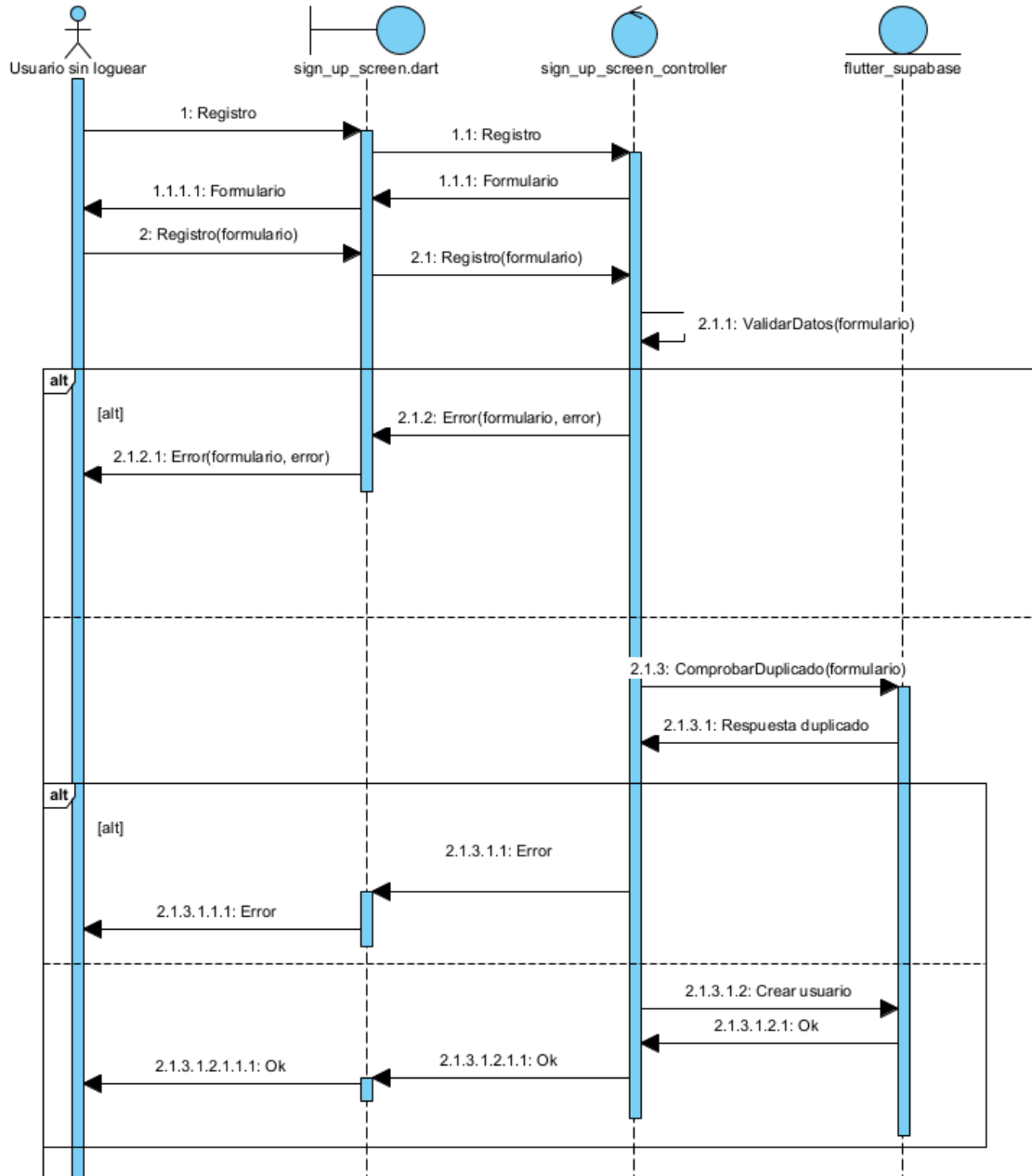


Figura 6: Diagrama de secuencia registro

Caso de uso **Login:**

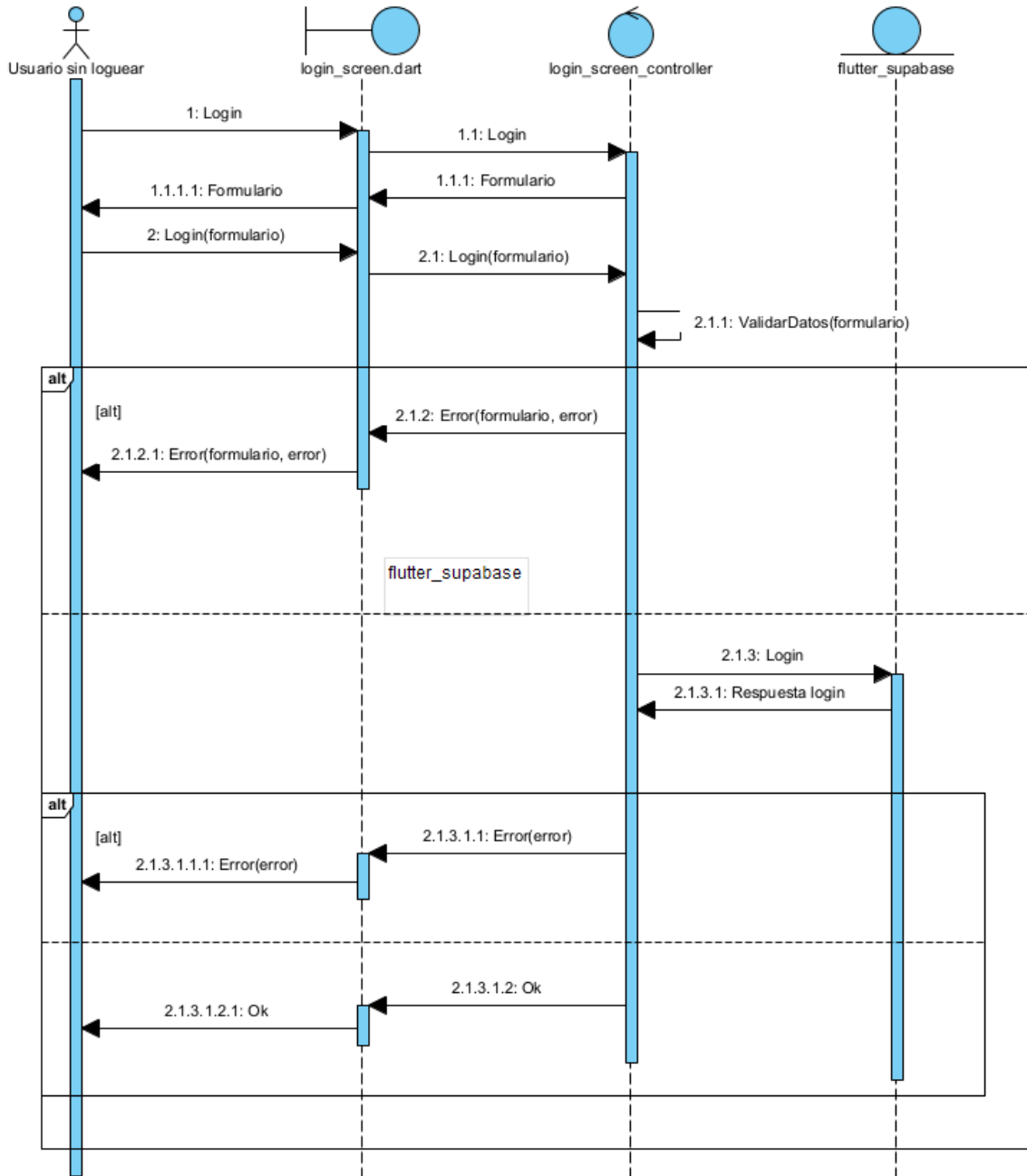


Figura 7: Diagrama de secuencia login

Anexo IV: Diseño del Sistema

Caso de uso **Logout:**

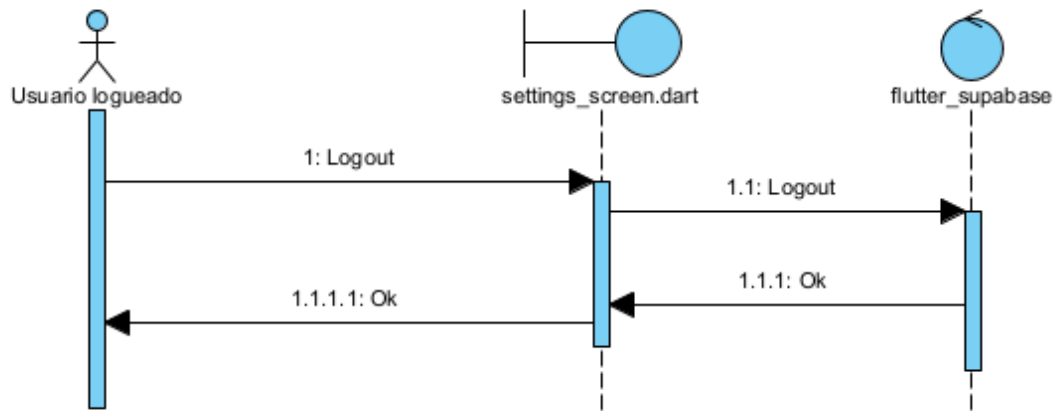


Figura 8: Diagrama de secuencia logout

Caso de uso **Recuperar contraseña:**

Anexo IV: Diseño del Sistema

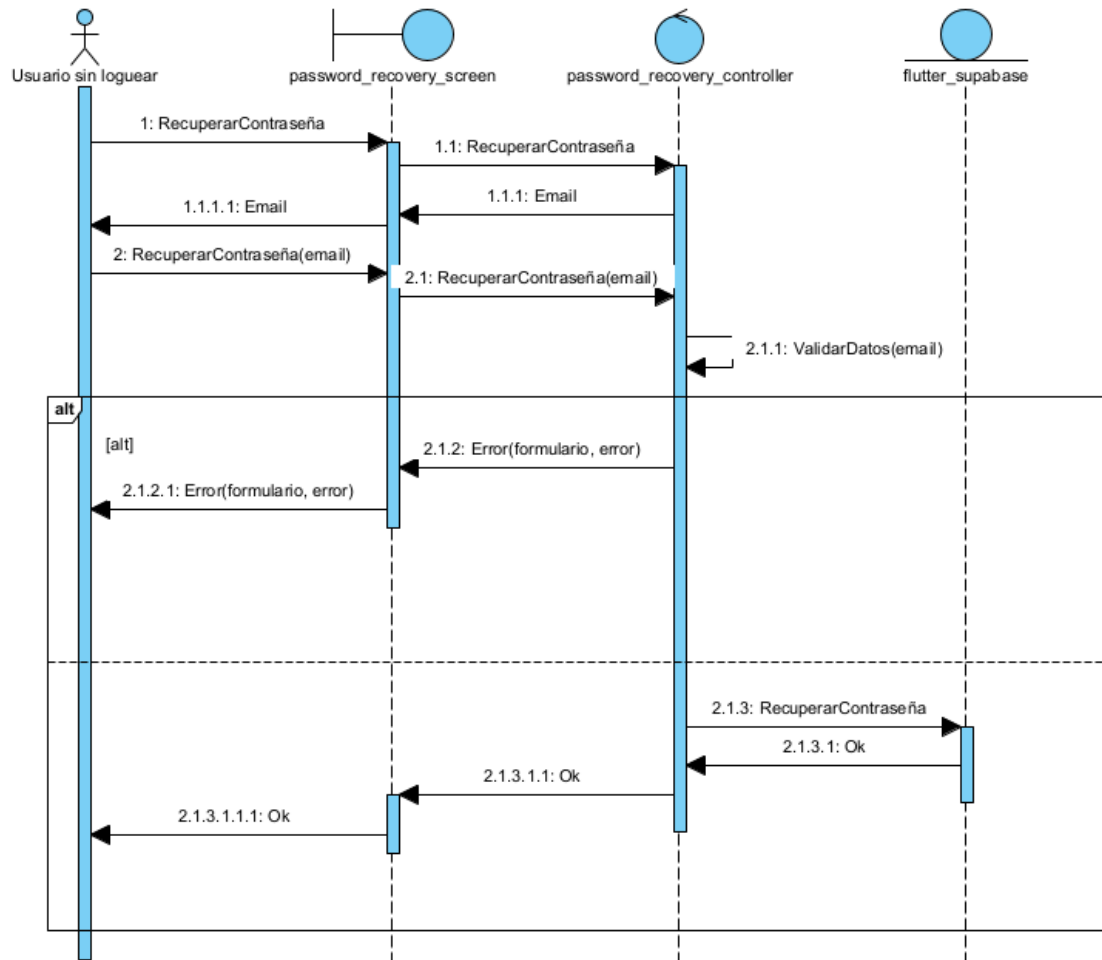


Figura 9: Diagrama de secuencia recuperar contraseña

3.4.2 Diagrama del paquete gestión usuarios Caso de uso Ver perfil:

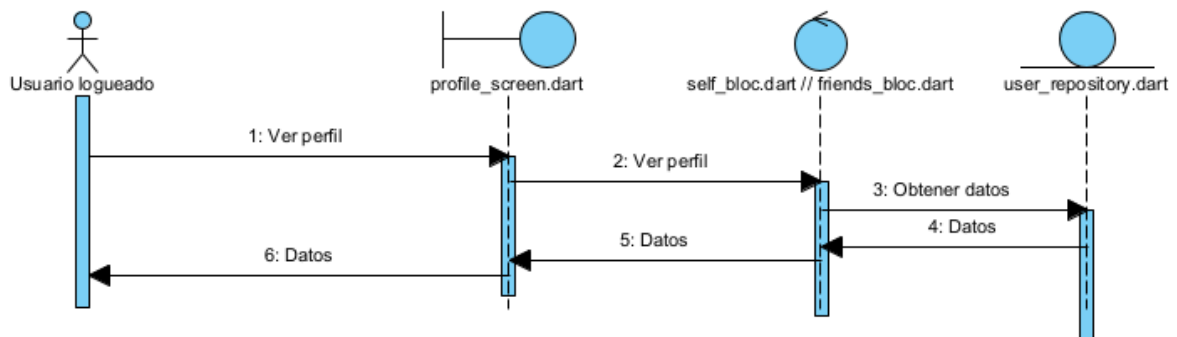


Figura 10: Diagrama de secuencia ver perfil

3.4.3 Diagrama del paquete gestión feed

Caso de uso **Crear publicación**:

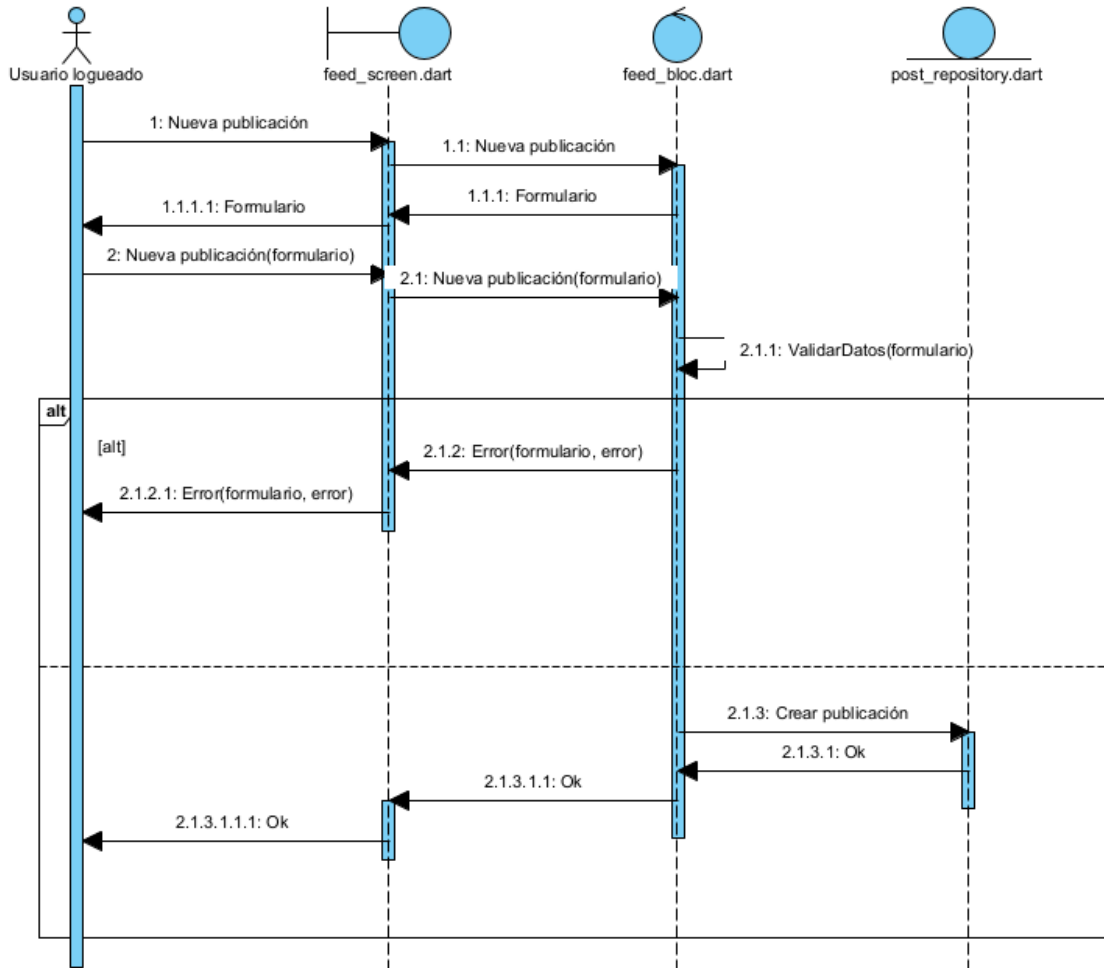


Figura 11: Diagrama de secuencia crear publicación

Caso de uso **Borrar publicación**:

Anexo IV: Diseño del Sistema

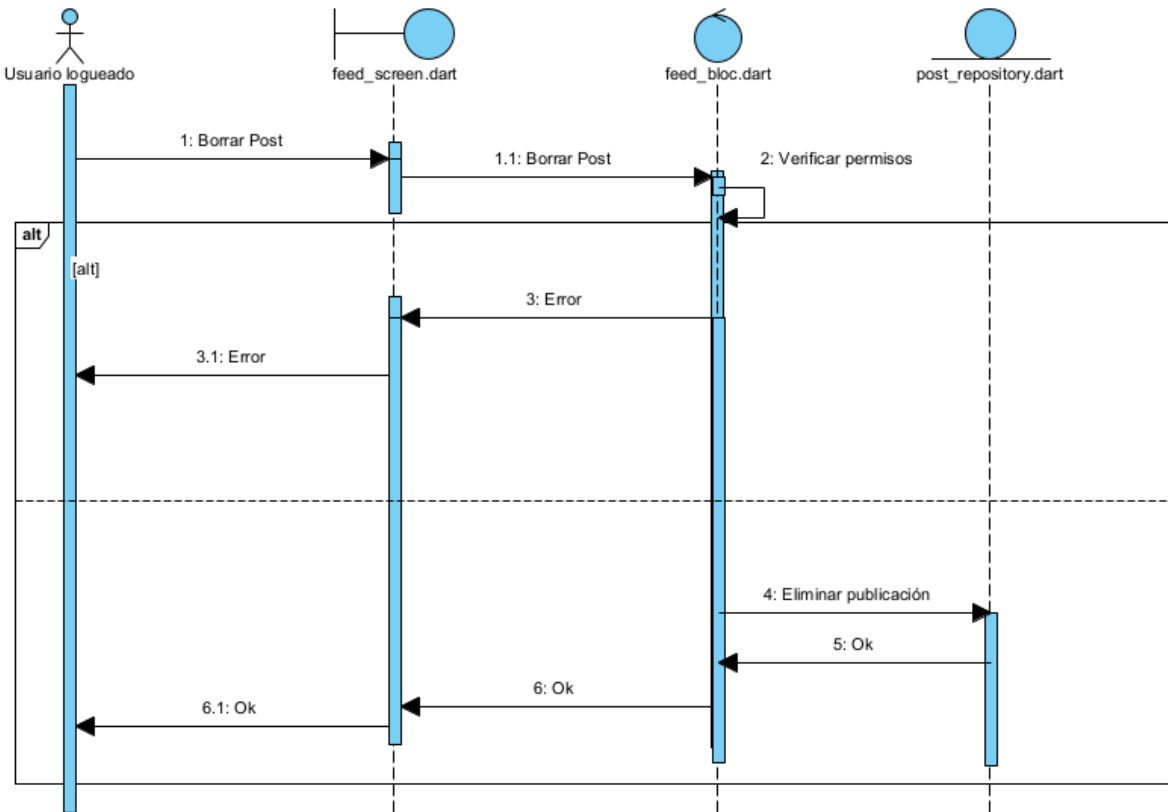


Figura 12: Diagrama de secuencia borrar publicación

Caso de uso Ver feed:

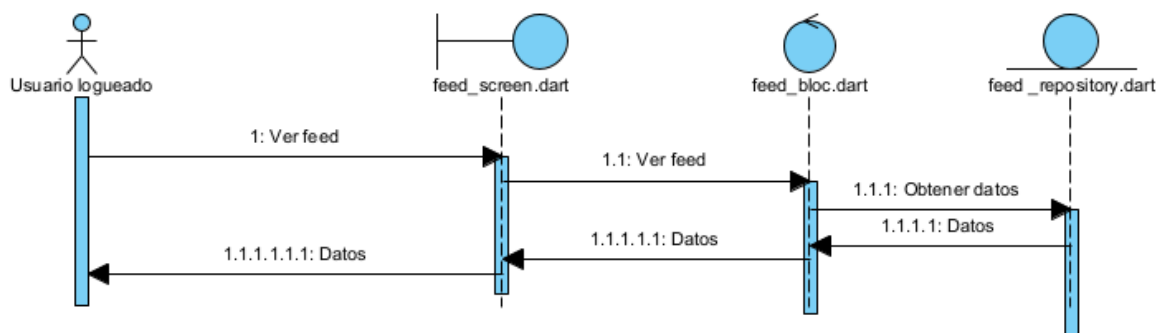


Figura 13: Diagrama de secuencia ver feed

Caso de uso Dar like:

Anexo IV: Diseño del Sistema

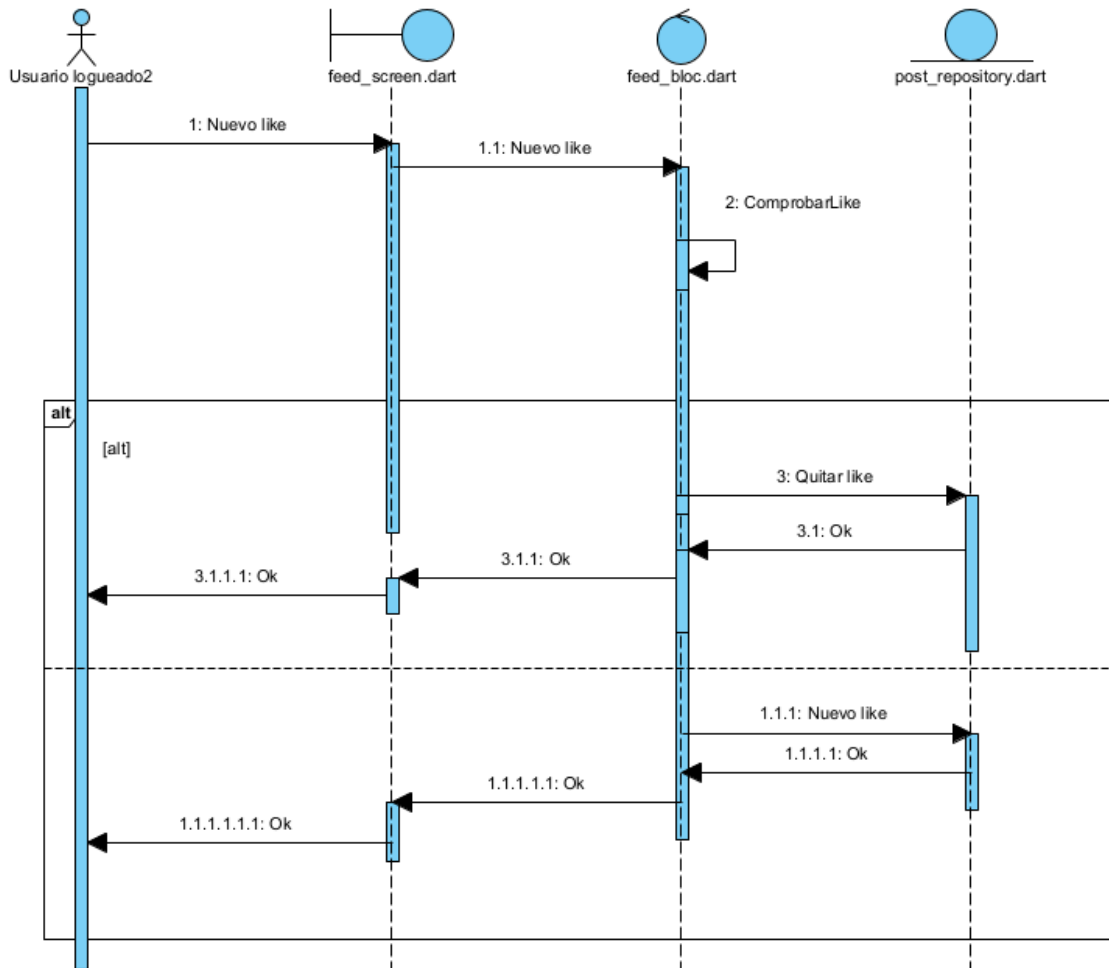


Figura 14: Diagrama de secuencia dar like

Caso de uso **Compartir publicación:**

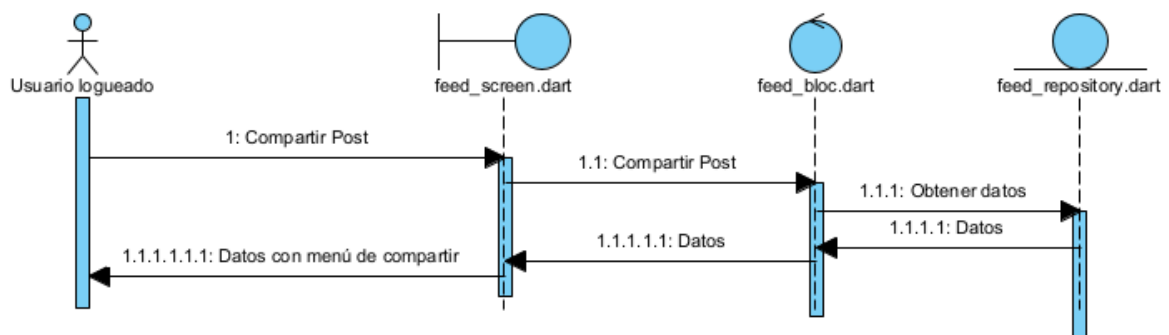


Figura 15: Diagrama de secuencia compartir publicación

3.4.4 Diagrama del paquete gestión social

Caso de uso **Enviar solicitud de amistad:**

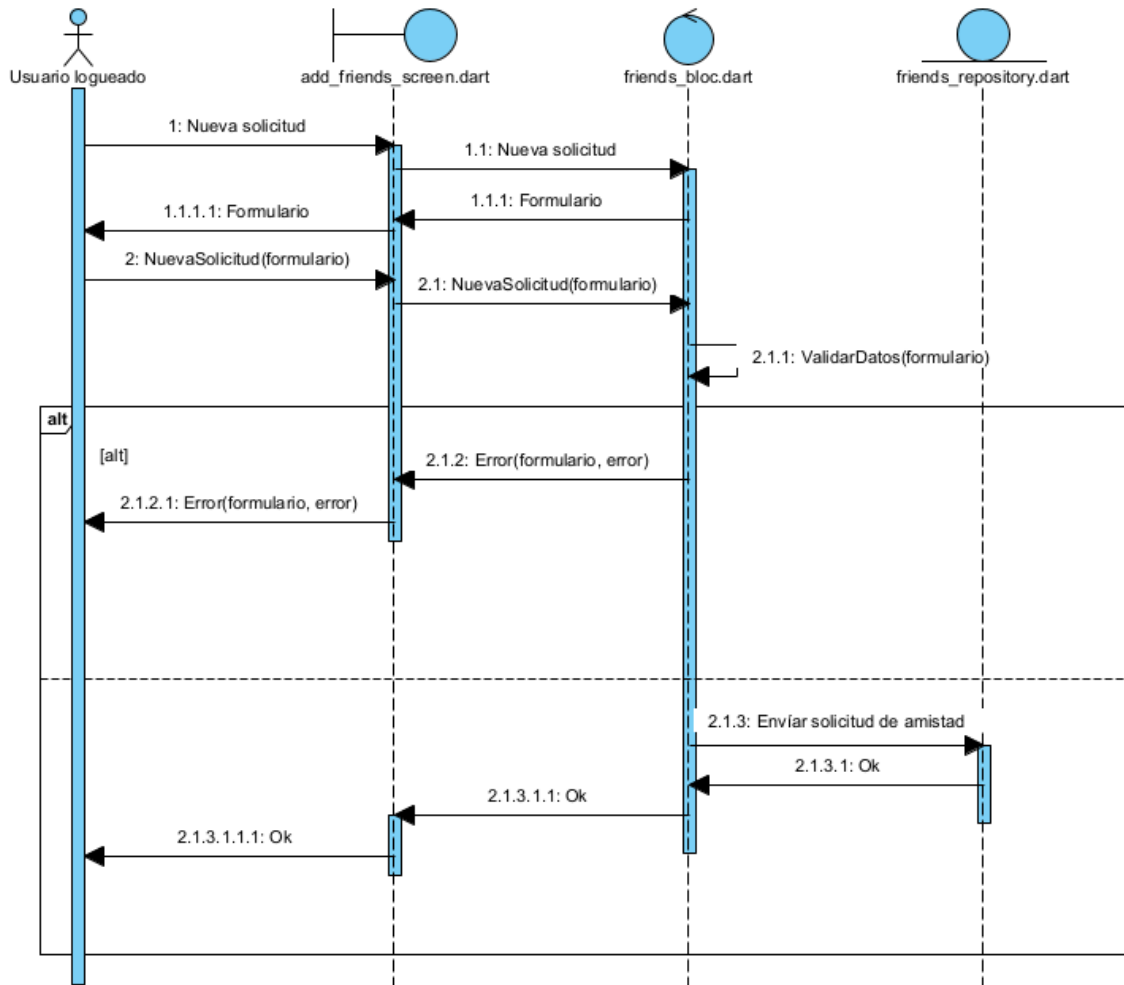


Figura 16: Diagrama de secuencia enviar solicitud de amistad

Caso de uso **Compartir código de amigos:**

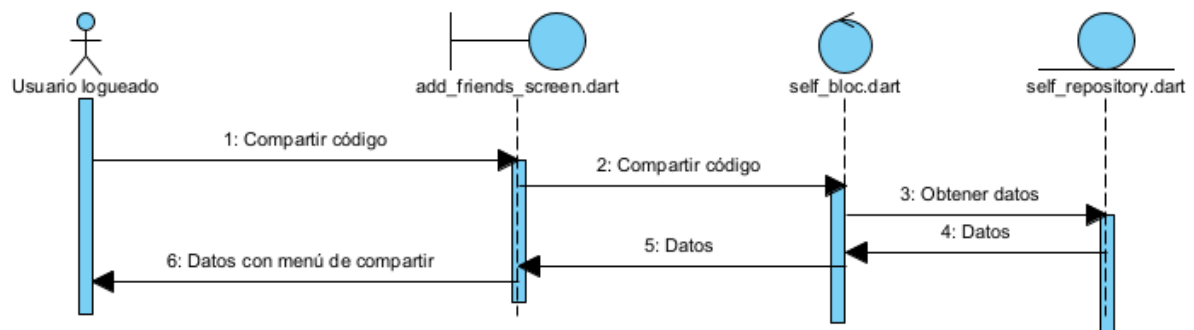


Figura 17: Diagrama de secuencia compartir código de amigos

Caso de uso **Ver solicitudes de amistad:**

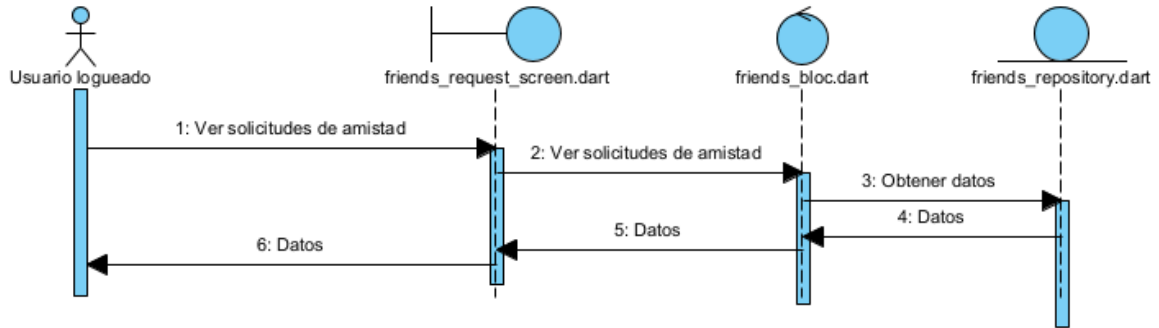


Figura 18: Diagrama de secuencia ver solicitudes de amistad

Caso de uso **Responder a la solicitud de amistad:**

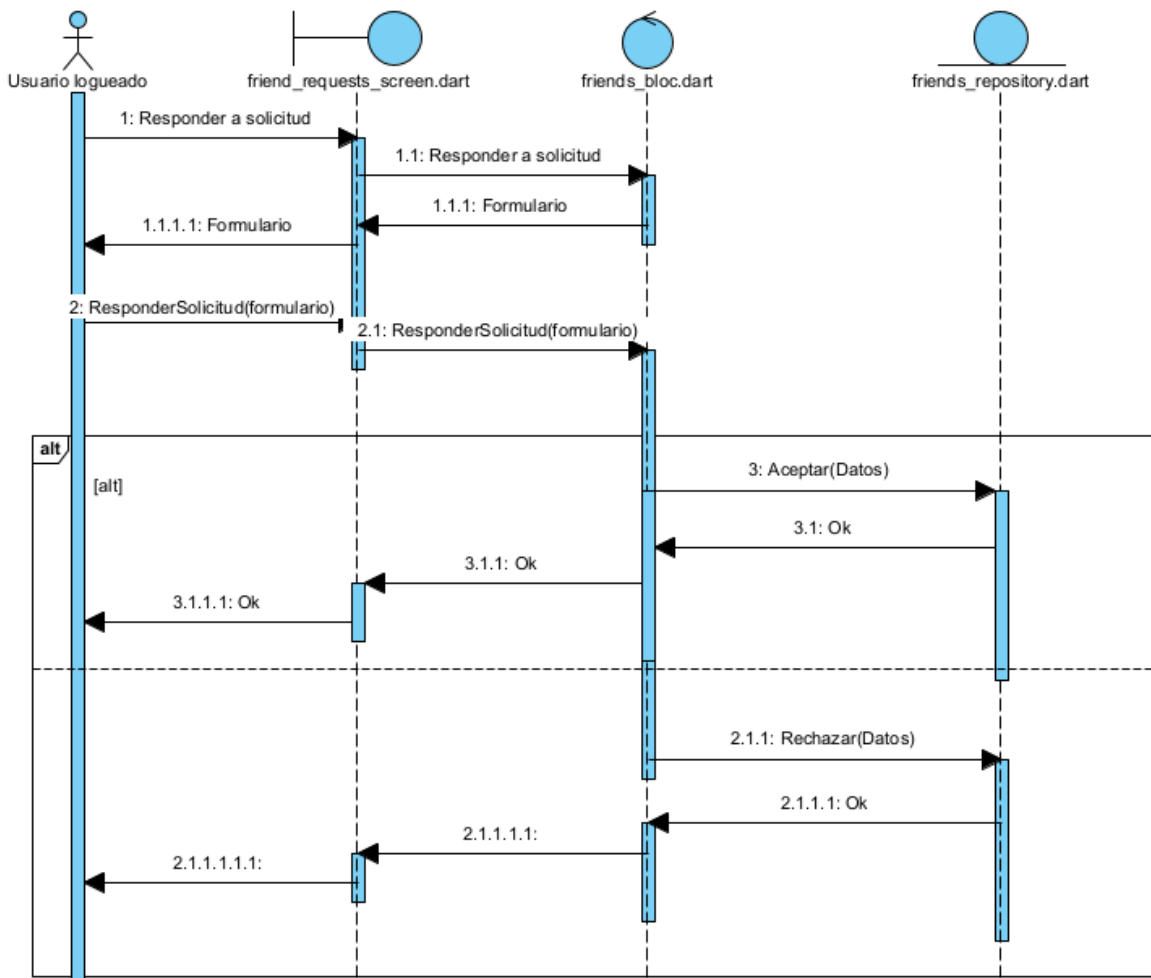


Figura 19: Diagrama de secuencia responder a petición de amistad

3.4.5 Diagrama del paquete gestión estadísticas

Caso de uso **Ver estadísticas**:

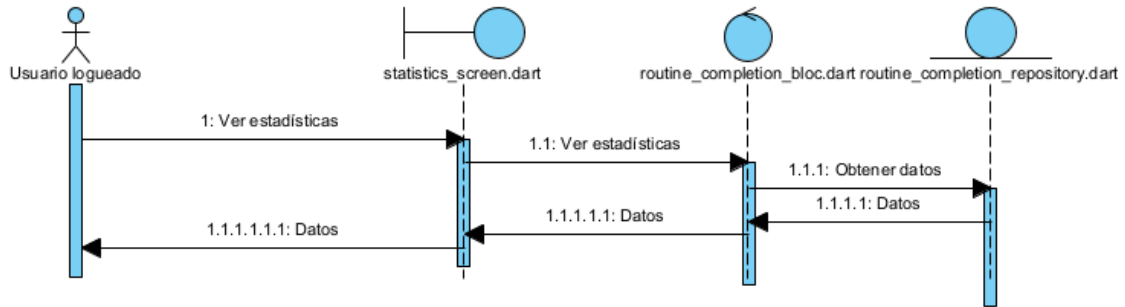


Figura 20: Diagrama de secuencia ver estadísticas

3.4.6 Diagrama del paquete gestión rutinas

Caso de uso **Crear rutina**:

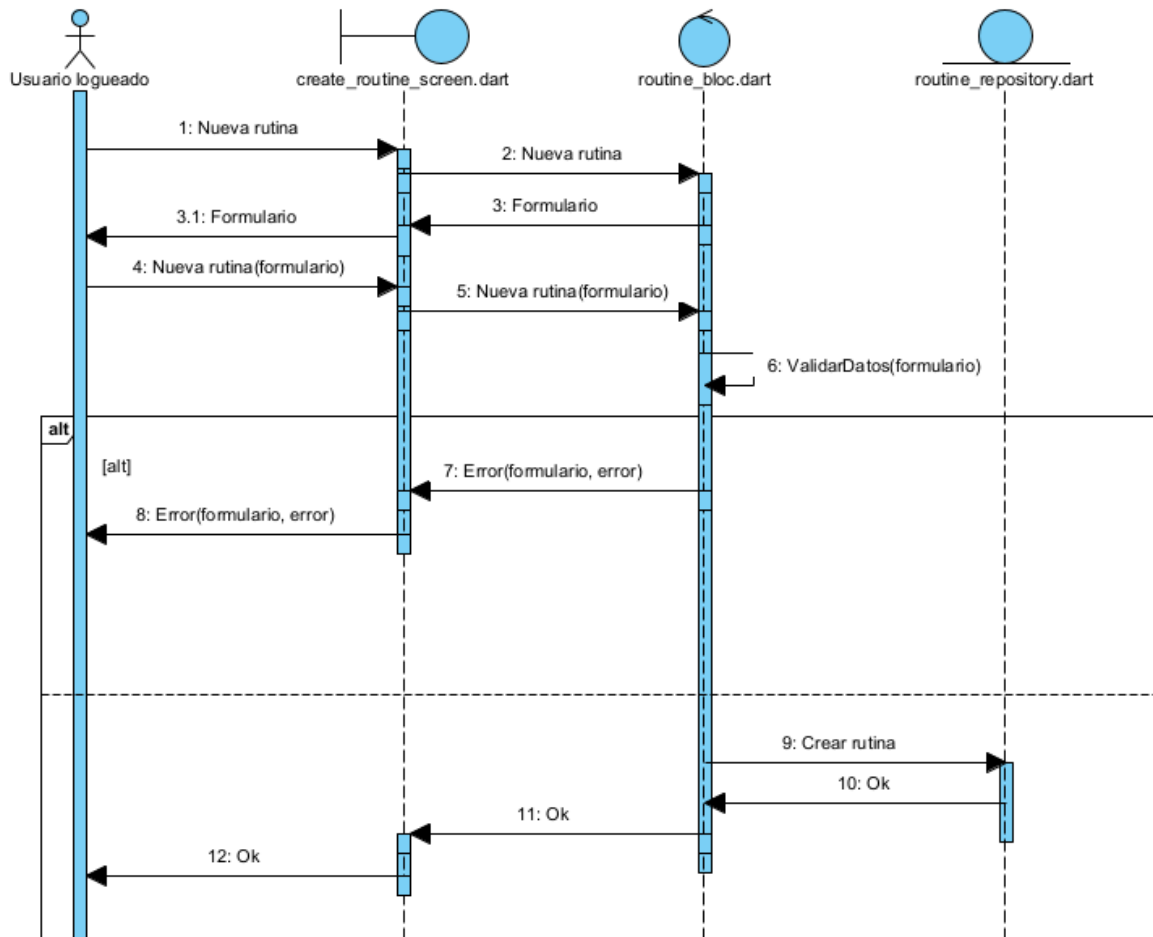


Figura 21: Diagrama de secuencia crear rutina

Caso de uso **Editar rutina:**

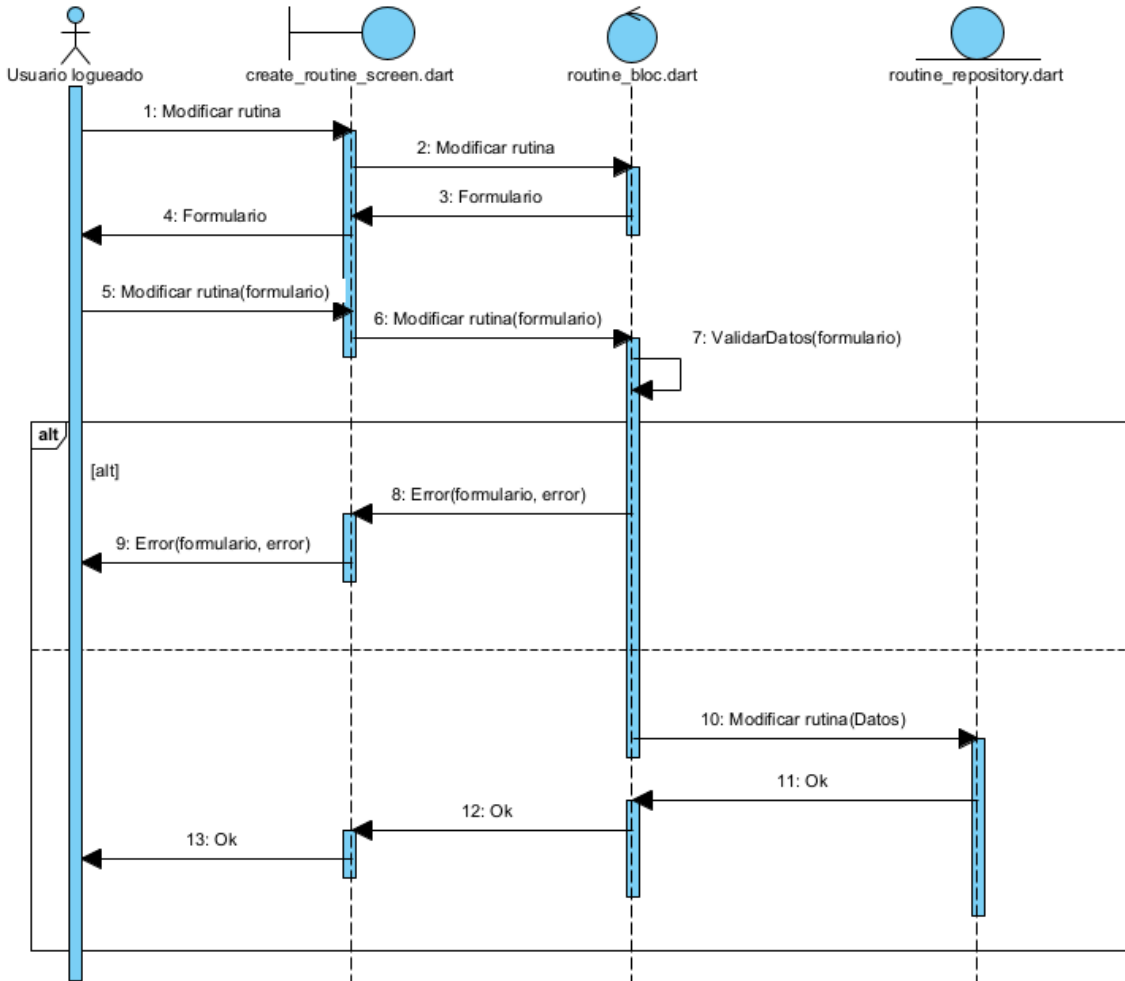


Figura 22: Diagrama de secuencia editar rutina

Caso de uso **Borrar rutina:**

Anexo IV: Diseño del Sistema

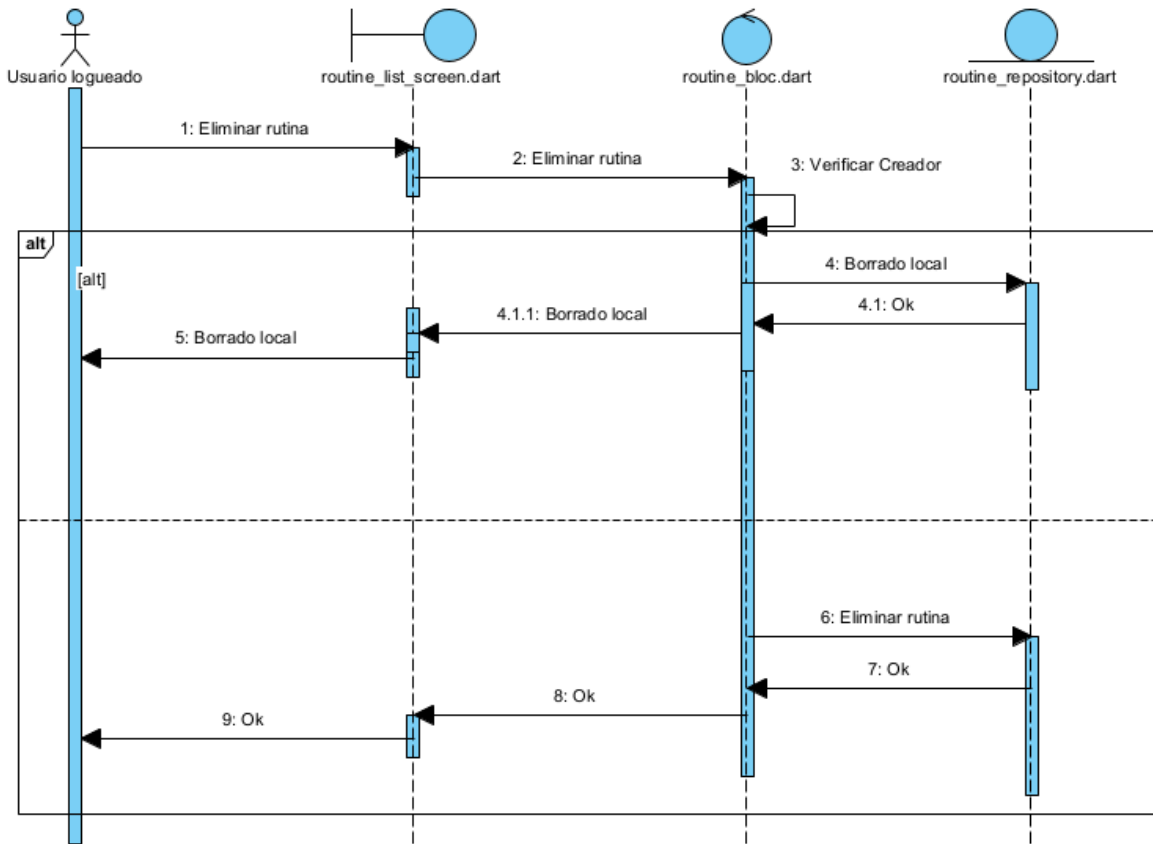


Figura 23: Diagrama de secuencia borrar rutina

Caso de uso Ver rutinas públicas:

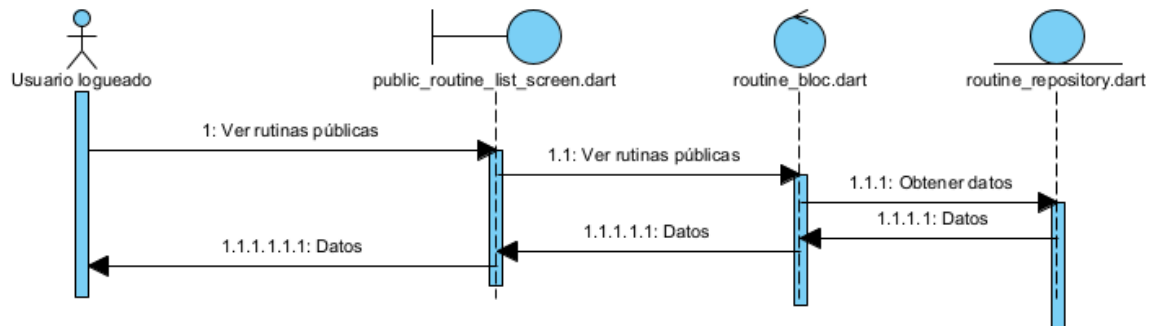


Figura 24: Diagrama de secuencia ver rutinas públicas

Caso de uso Añadir rutina pública:

Anexo IV: Diseño del Sistema

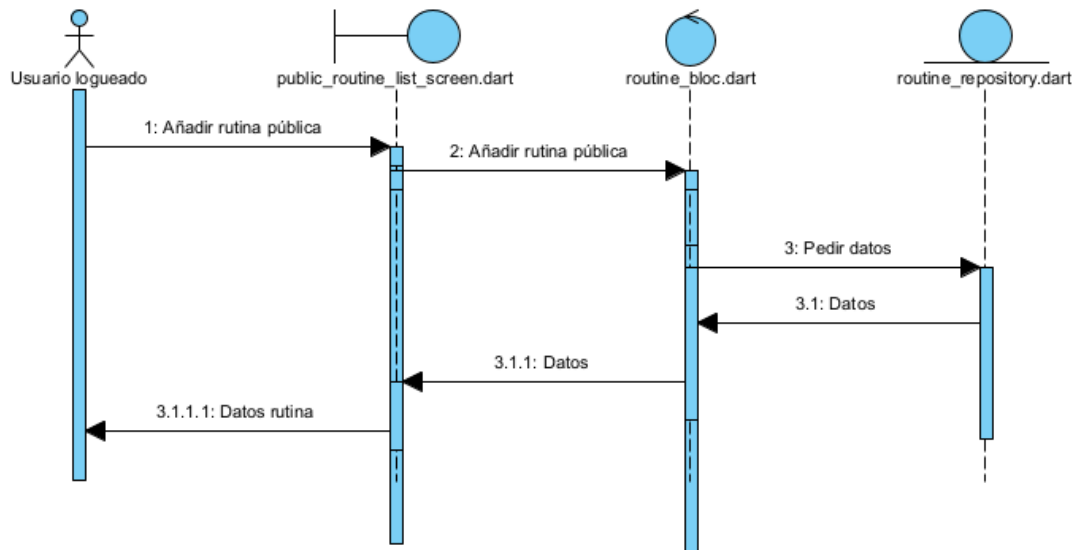


Figura 25: Diagrama de secuencia añadir rutina pública

Caso de uso **Completar rutina:**

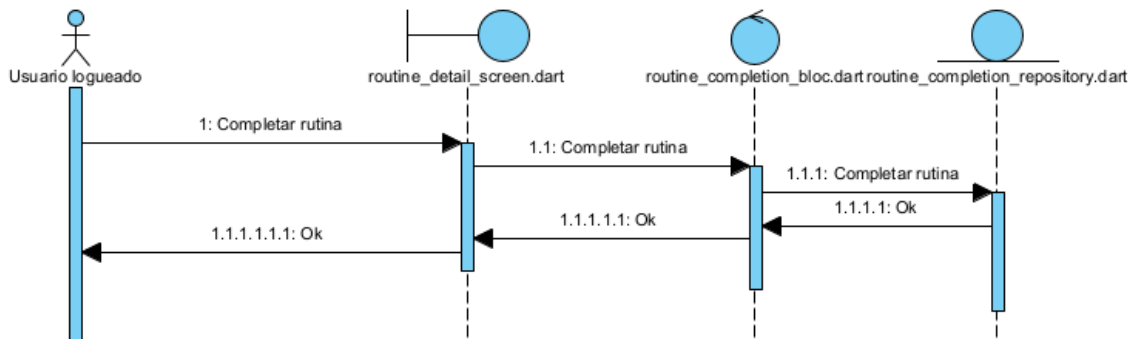


Figura 26: Diagrama de secuencia completar rutina

3.4.7 Diagrama del paquete gestión logros

Caso de uso **Desbloquear logro:**

Anexo IV: Diseño del Sistema

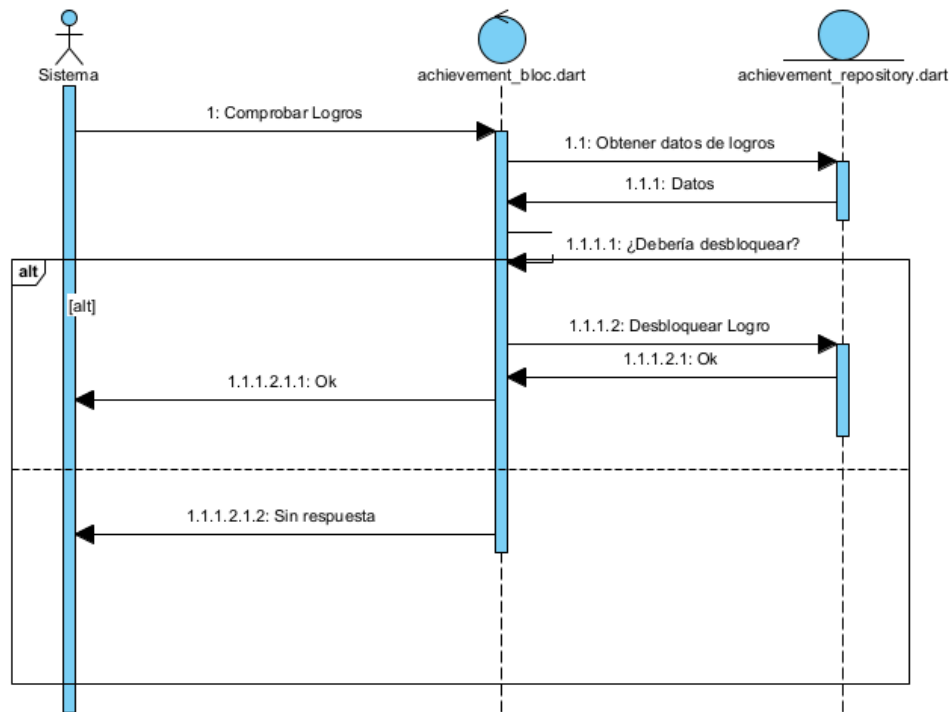


Figura 27: Diagrama de secuencia desbloquear logro

Caso de uso Ver logros:

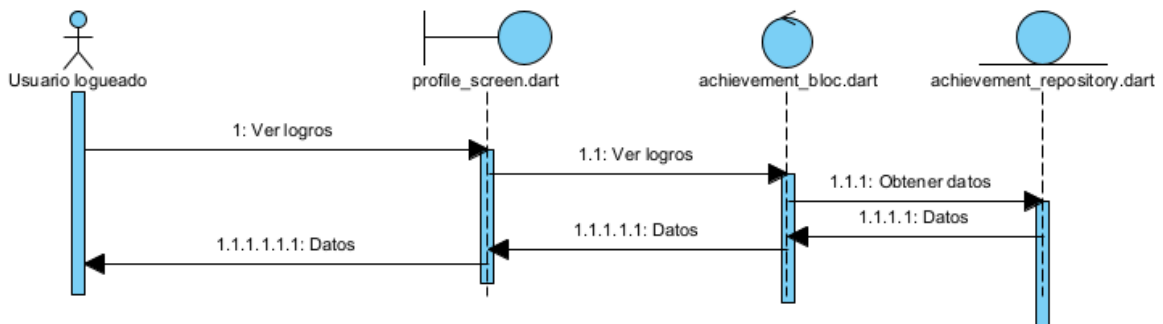


Figura 28: Diagrama de secuencia ver logros

3.4.8 Diagrama del paquete gestión notificaciones:

Diagrama de uso Programar notificaciones:

Anexo IV: Diseño del Sistema

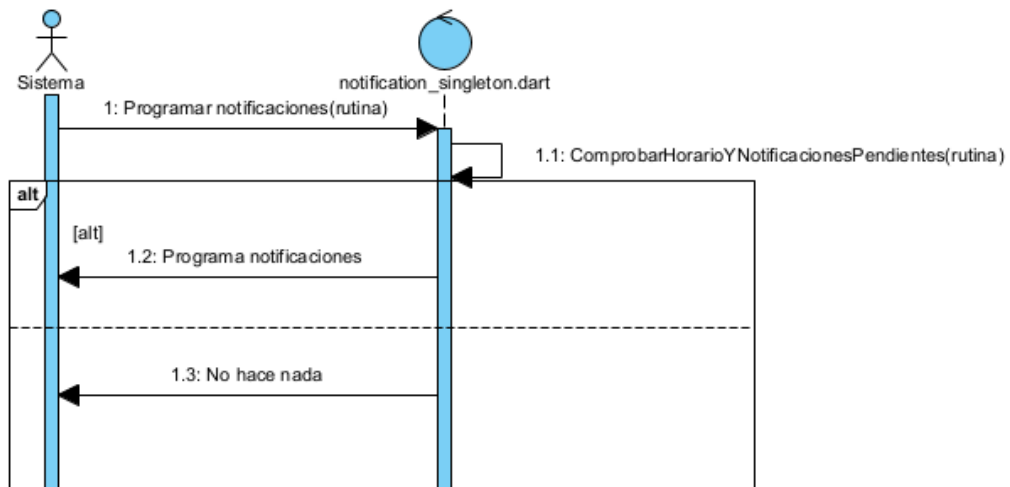


Figura 29: Diagrama de secuencia programar notificaciones

3.4.9 Diagrama del paquete gestión juego

Caso de uso **Ver juego:**

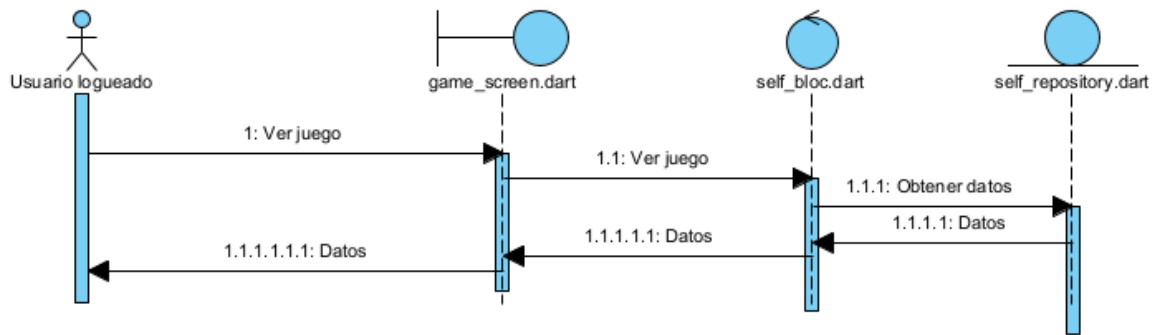


Figura 30: Diagrama de secuencia ver juego

Caso de uso **Editar flores:**

Anexo IV: Diseño del Sistema

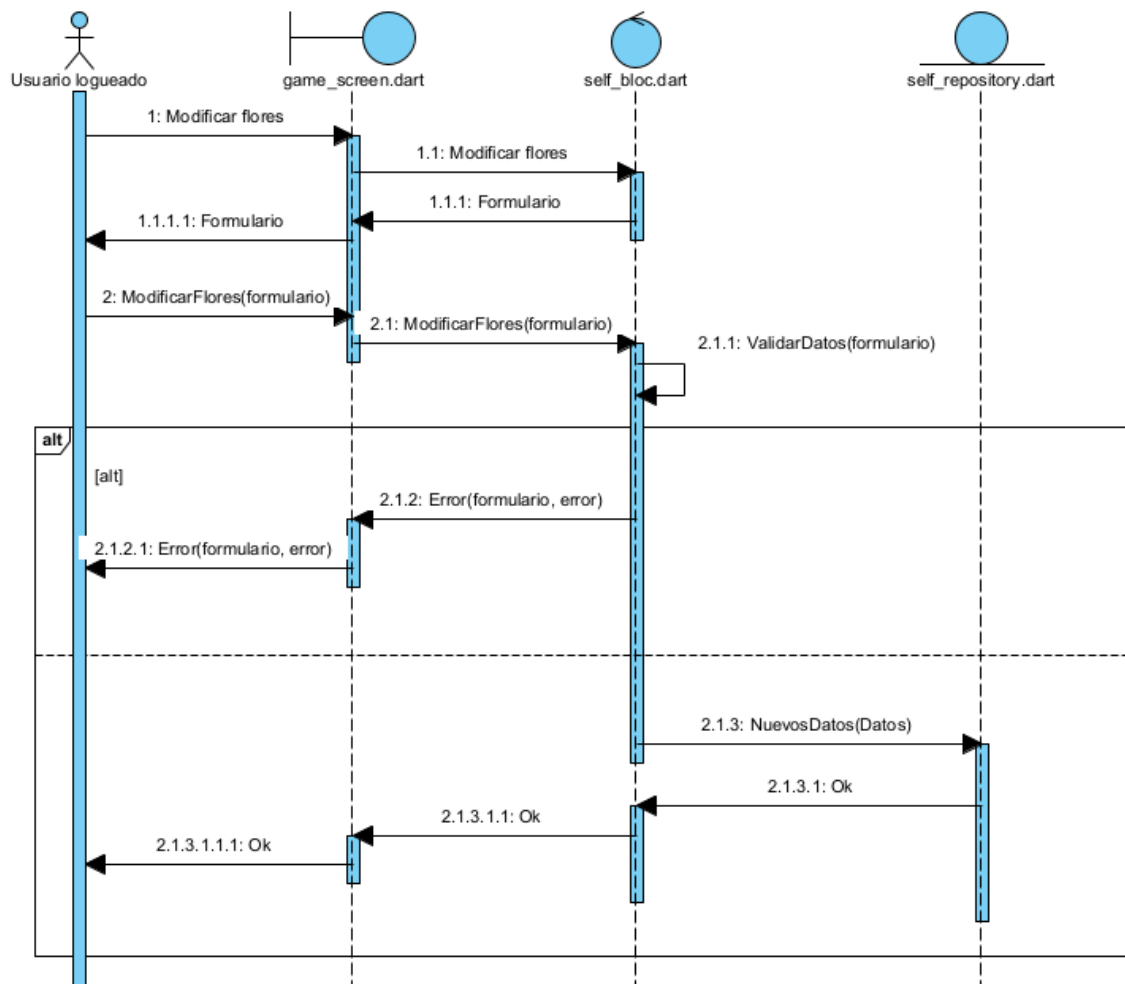


Figura 31: Diagrama de secuencia cambiar flores

5 MODELO DE DESPLIEGUE

En la siguiente figura se puede apreciar el diagrama de despliegue del sistema diseñado. En el despliegue tenemos 3 nodos:

- **Un teléfono Android:** Representa cualquier dispositivo que pueda ejecutar el artefacto de la aplicación para Android.
- **Un teléfono iOS:** Representa cualquier dispositivo que pueda ejecutar el artefacto de la aplicación para iOS.
- **Supabase:** Nuestra base de datos elegida para el desarrollo del proyecto. Ambos artefactos que hemos explicado previamente se conectarán con ella a través de internet, para obtener los datos requeridos para la ejecución.

Las comunicaciones entre los nodos del sistema se realizan a través de internet.

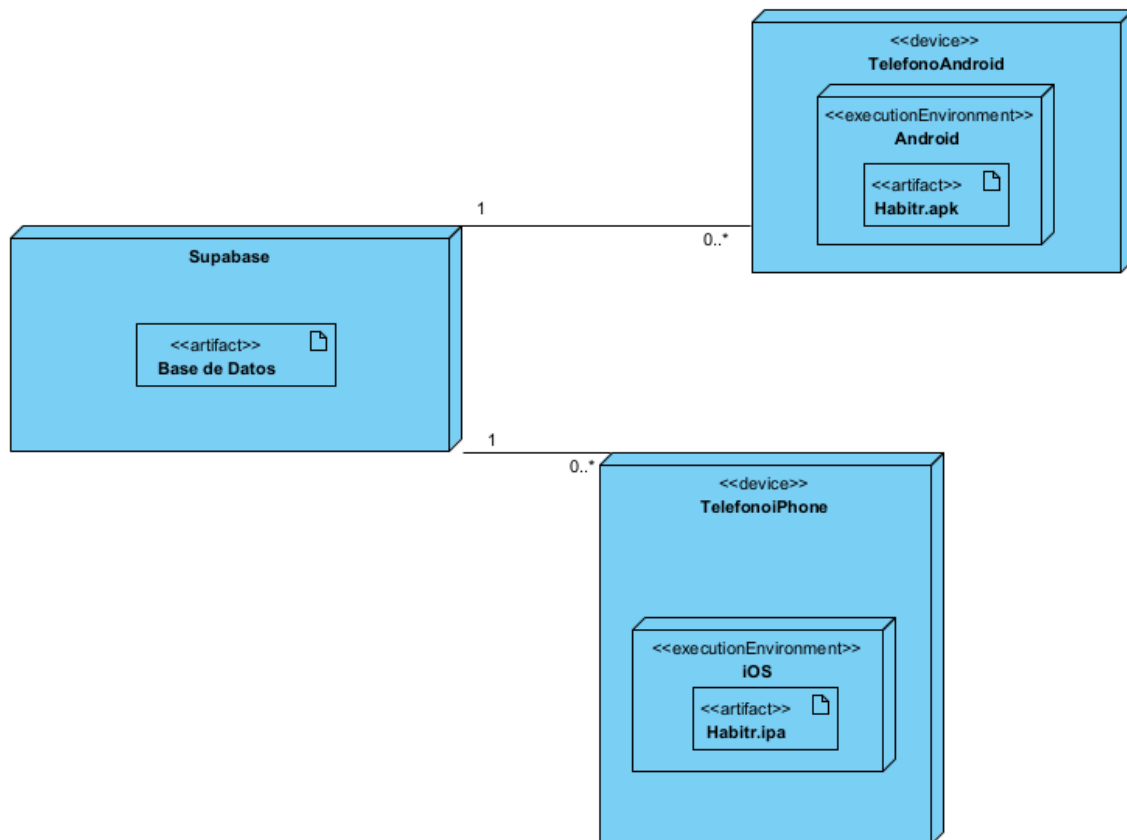


Figura 33: Diagrama de despliegue

6 REFERENCIAS

- Pressman, R. S. - "Ingeniería del Software: Un Enfoque Práctico". 7ª Edición. McGraw-Hill. 2010.