**Turnosync: Aplicación android para la gestión de turnos laborales**

Trabajo de Fin de Grado

GRADO EN INGENIERÍA INFORMÁTICA

**Anexo III - Especificación de Diseño**



Septiembre de 2018

Autor

Julio García Valdunciel

Tutor

Iván Álvarez Navia

Tabla de contenido

[1. Introducción 7](#_Toc523158633)

[2. Análisis 8](#_Toc523158634)

[2.1. • Diagrama de paquetes 8](#_Toc523158635)

[2.2. Modelo de dominio 9](#_Toc523158636)

[2.3. Diagrama de paquetes de análisis. 10](#_Toc523158637)

[2.4. Diagramas de interacción 11](#_Toc523158638)

[3. Diseño 16](#_Toc523158639)

[3.1. Arquitectura 16](#_Toc523158640)

[3.2. Diagrama de despliegue 18](#_Toc523158641)

[3.3. Diseño de la base de datos 19](#_Toc523158642)

Tabla de ilustraciones

[Ilustración 1 - Diagrama de paquetes 8](#_Toc523158643)

[Ilustración 2 - Modelo de dominio 9](#_Toc523158644)

[Ilustración 3 - Diagrama de paquetes de análisis 10](#_Toc523158645)

[Ilustración 4 - Diagrama de secuencia Iniciar sesión 11](#_Toc523158646)

[Ilustración 5 - Diagrama de secuencia Editar calendario 11](#_Toc523158647)

[Ilustración 6 - Diagrama de secuencia Crear grupo 12](#_Toc523158648)

[Ilustración 7 - Diagrama de secuencia Mostrar tipos de turno 12](#_Toc523158649)

[Ilustración 8 - Diagrama de secuencia Añadir tipo de turno 13](#_Toc523158650)

[Ilustración 9 - Diagrama de secuencia Añadir turnos 13](#_Toc523158651)

[Ilustración 10 - Diagrama de secuencia Establecer horas máximas semanales 14](#_Toc523158652)

[Ilustración 11 - Diagrama de secuencia Invitar usuario 14](#_Toc523158653)

[Ilustración 12 - Diagrama de secuencia Actualizar Token 14](#_Toc523158654)

[Ilustración 13 - Diagrama de secuencia Solicitar cambio de turno 15](#_Toc523158655)

[Ilustración 14 - Diagrama de secuencia Aceptar solicitud de cambio de turno 15](#_Toc523158656)

[Ilustración 15 - Diagrama de arquitectura MVC 17](#_Toc523158657)

[Ilustración 16 - Diagrama de despliegue 18](#_Toc523158658)

[Ilustración 17 - Diseño de base de datos 19](#_Toc523158659)

# Introducción

Este anexo recoge la documentación referente a la etapa de Análsis y Diseño de la ingeniería del software. A partir de los objetivos y requisitos del sistema contemplados en más detalle en el Anexo II – Especificación de Requisitos del Software se obtienen una serie de modelos centrados en el dominio de la solución y realizando aproximaciones a la implementación.

Como consecuencia labor de Análisis se han conseguido los siguientes resultados:

* **Diagrama de paquetes**: Se ha realizado un diagrama de paquetes divididos por las diferentes secciones de la aplicación.
* **Modelo de dominio**: Se ha realizado un modelado del dominio del sistema mostrando la interacción entre los distintos objetos.
* **Diagrama de paquetes de análisis**: Se ha realizado un diagrama de análisis donde se muestra la interacción entre los distintos componentes del sistema separados por paquetes. Es la primera aproximación las clases finales que se van a implementar.
* **Diagramas de interacción**: Se realizan los casos de uso mostrando la interacción entre los componentes del sistema necesarios para llevar a cabo el caso de uso.

Como consecuencia de la labor de Diseño se han conseguido los siguientes resultados:

* **Diagrama de arquitectura**: Se ha realizado un diagrama de arquitectura aplicando el patrón MVC al sistema y diagramas realizados en la fase de análisis.
* **Modelo de despliegue**: Se ha realizado un diagrama de despliegue donde se refleja la relación entre la aplicación y la plataforma Firebase.
* **Diseño de la base de datos**: Se indica como se ha construido la base de datos no relacional mediante un diagrama que muestra la jerarquía de objetos.

# Análisis

Utilizando los objetivos y requisitos definidos en la especificación de requisitos se modela el sistema para empezar a definir su estructura.

## Diagrama de paquetes

Se separa el sistema en paquetes donde cada uno encapsula un conjunto de funcionalidades del sistema relacionadas entre sí.

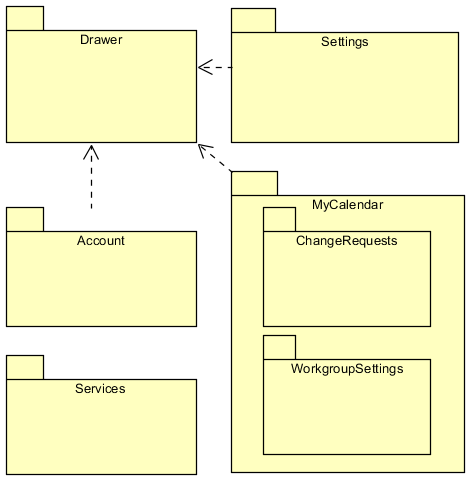


Ilustración - Diagrama de paquetes

* **Account**: Contiene la funcionalidad relacionada con la gestión de usuarios siendo estas el registro, inicio de sesión y recuperación de contraseña.
* **Drawer**: Contiene la funcionalidad relacionada con la navegación principal de la aplicación que se compone del panel lateral y la pantalla principal con la creación de grupos de trabajo.
* **Settings**: Contiene la funcionalidad relacionada con las opciones de la aplicación que contiene la gestión de la cuenta del usuario.
* **MyCalendar**: Contiene la funcionalidad relacionada con las pantallas de calendarios encargadas de mostrar la vista del calendario personal y del calendario grupal, manejando las modificaciones en el calendario por parte del coordinador.
* **WorkgroupSettings**: Contiene la funcionalidad relacionada con las opciones del grupo de trabajo, siendo estas la gestión de los usuarios del grupo de trabajo, la gestión de los tipos de turnos y del número máximo de horas semanales.
* **ChangeRequests**: Contiene la funcionalidad relacionada con los cambios de grupo donde es posible visualizar aceptar o rechazar cambios de grupo.

## Modelo de dominio

Se ha realizado el modelado de dominio donde se muestran la relación entre los objetos del sistema.

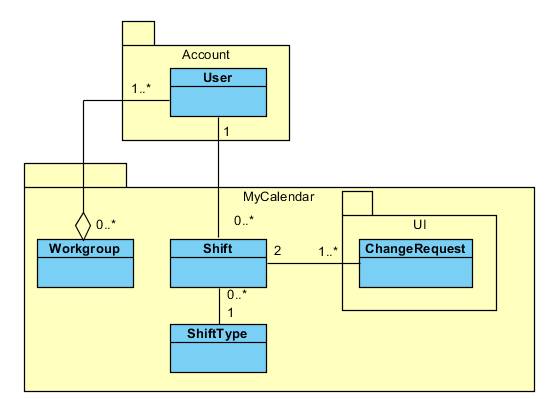


Ilustración - Modelo de dominio

* **User**: Representa a un usuario del sistema.
* **Workgroup**: Representa a un grupo de trabajo compuesto por usuarios.
* **Shift**: Representa a un turno de un usuario dentro de un grupo de trabajo.
* **ShiftType** Representa las características de un turno.
* **ChangeRequest**: Representa la solicitud de un cambio de turno entre usuarios.

## Diagrama de paquetes de análisis.

Se realiza la primera aproximación del sistema con los distintos componentes del sistema, empezando a dividir las vistas, controladores y datos indicando como se relacionan entre ellos.

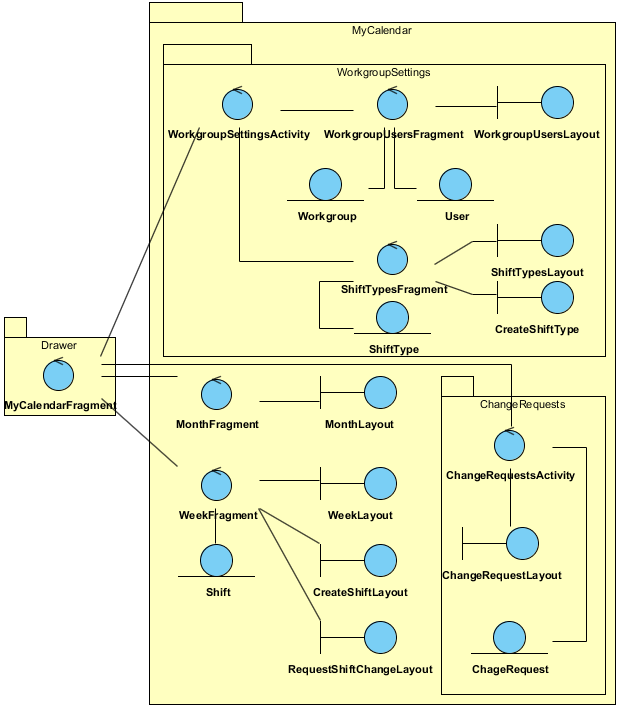


Ilustración - Diagrama de paquetes de análisis

## Diagramas de interacción

Se han realizado los casos de uso mediante diagramas de interacción indicando cómo interactúan los actores para conseguir la meta de cada caso de uso. A continuación los diagramas más importantes:

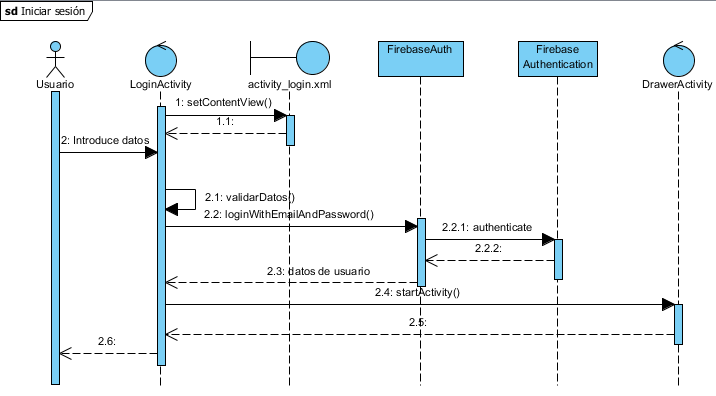


Ilustración - Diagrama de secuencia Iniciar sesión

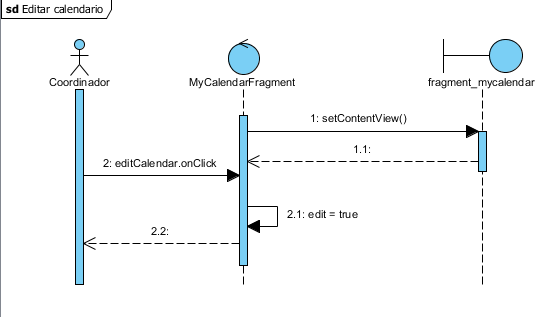


Ilustración - Diagrama de secuencia Editar calendario

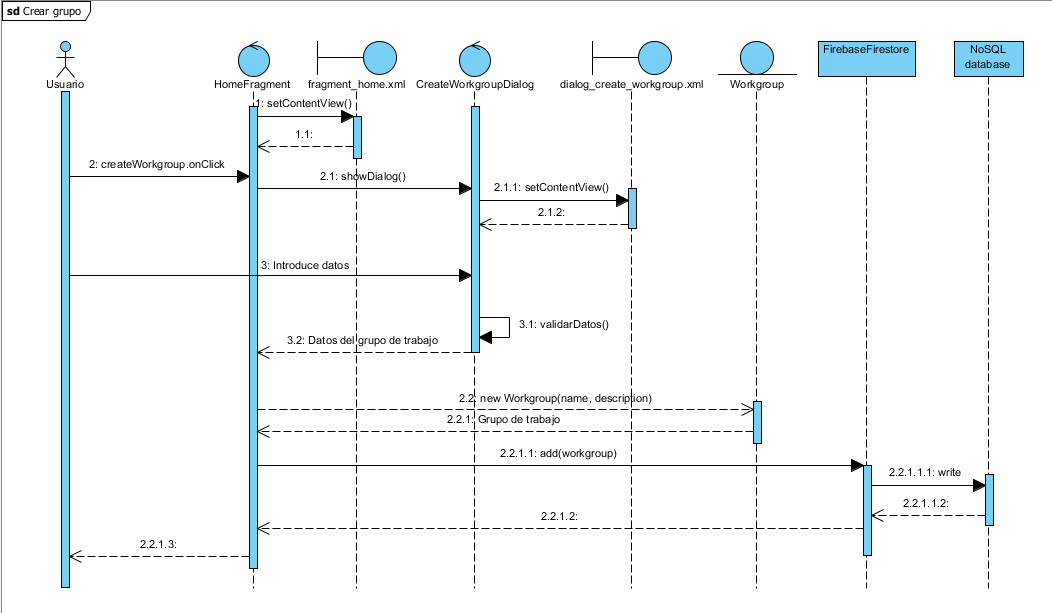


Ilustración - Diagrama de secuencia Crear grupo

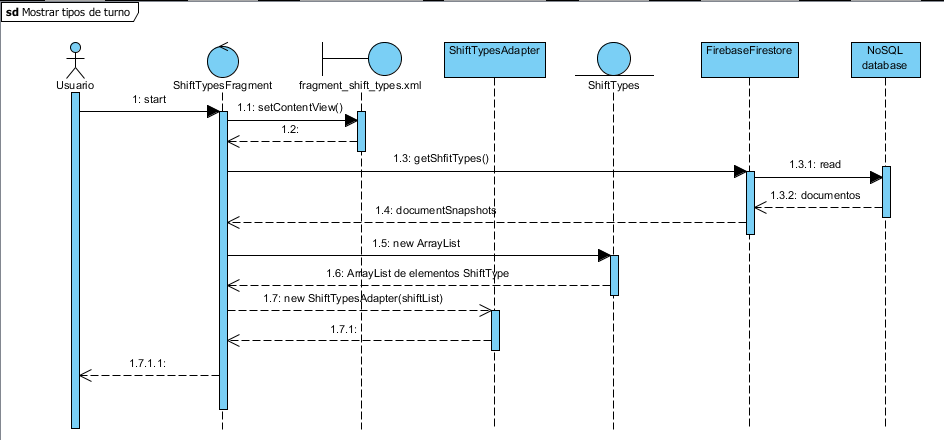


Ilustración - Diagrama de secuencia Mostrar tipos de turno

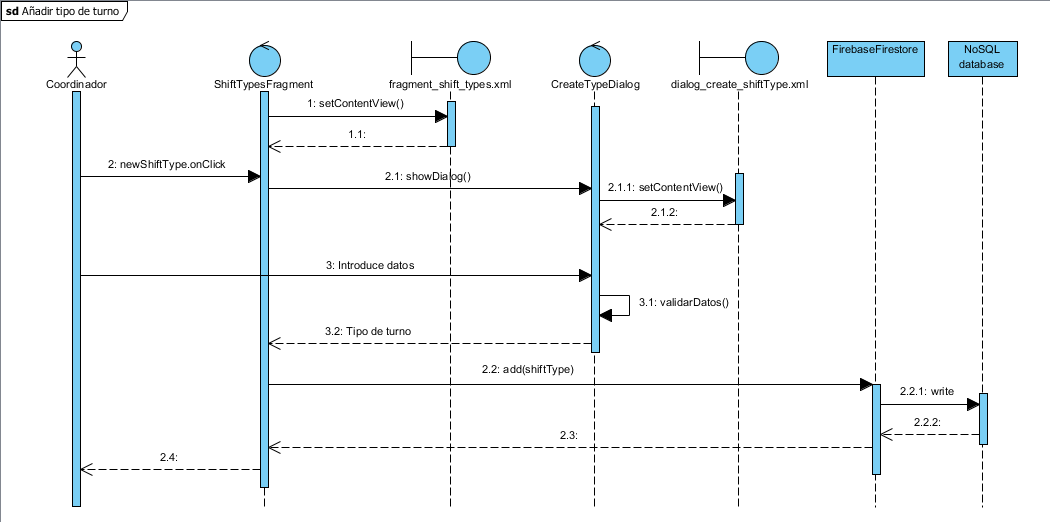


Ilustración - Diagrama de secuencia Añadir tipo de turno

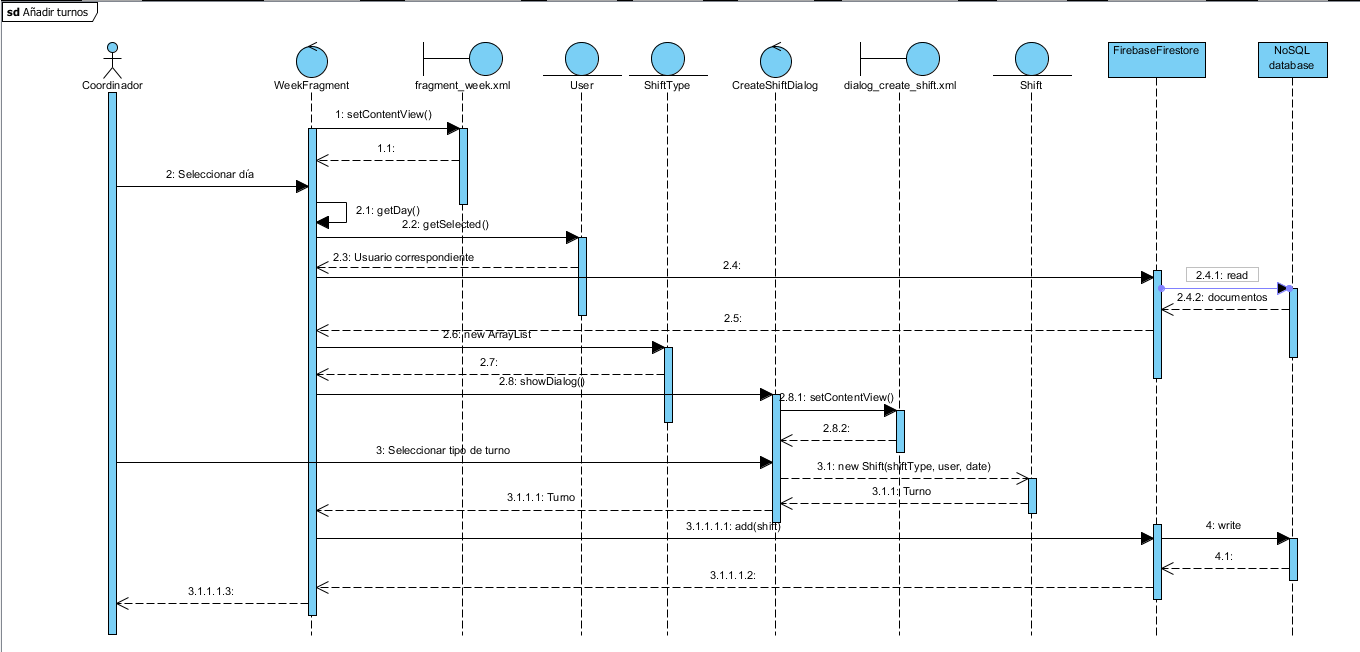


Ilustración - Diagrama de secuencia Añadir turnos

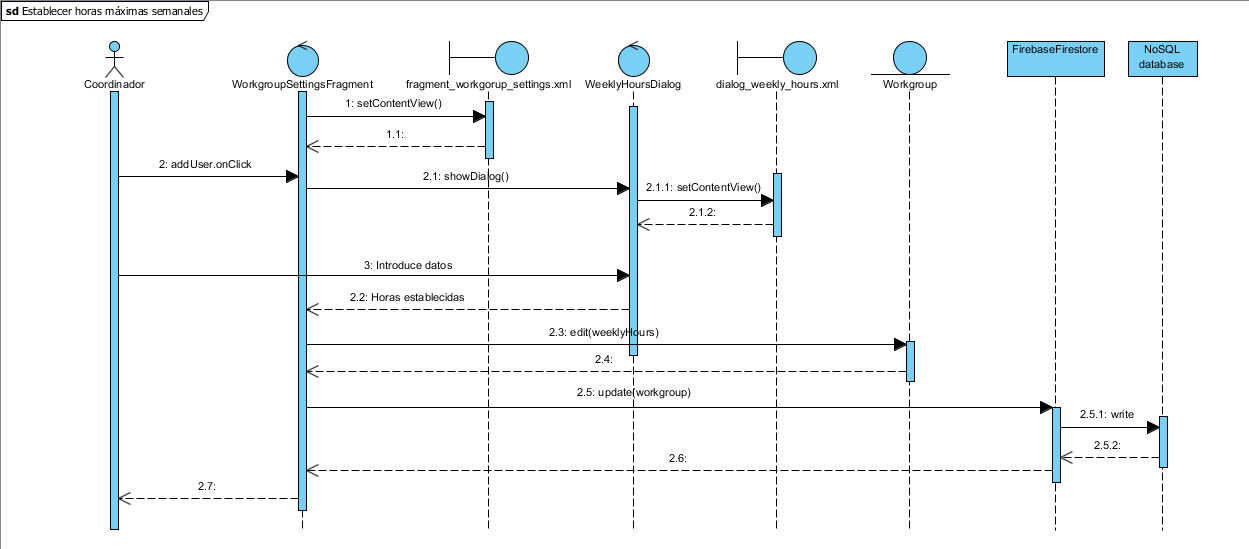


Ilustración - Diagrama de secuencia Establecer horas máximas semanales

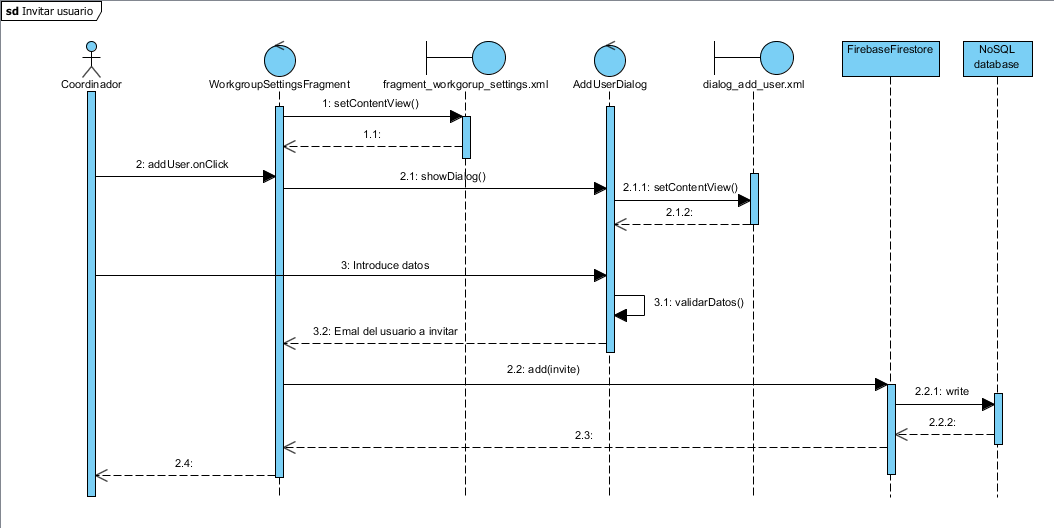


Ilustración - Diagrama de secuencia Invitar usuario

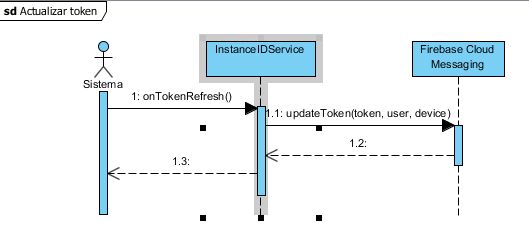


Ilustración - Diagrama de secuencia Actualizar Token

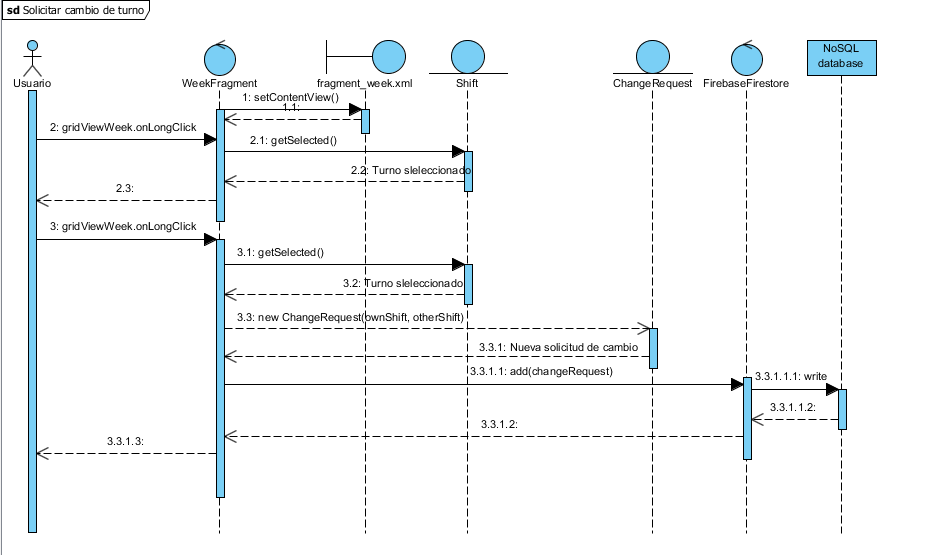


Ilustración - Diagrama de secuencia Solicitar cambio de turno

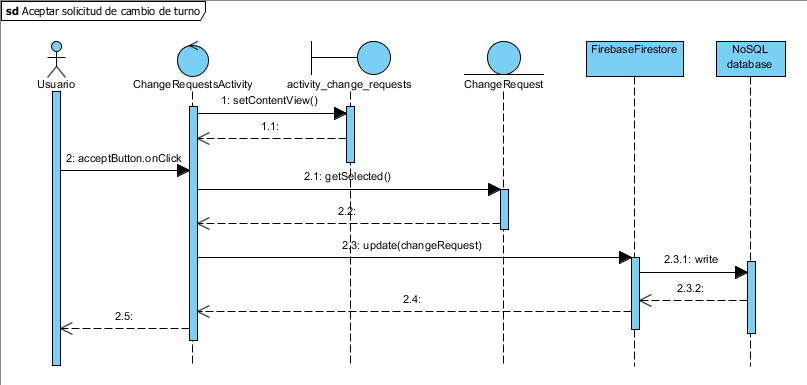


Ilustración - Diagrama de secuencia Aceptar solicitud de cambio de turno

# Diseño

En la fase de diseño se ha utilizado una arquitectura MVC (Model, View, Controller o Modelo, Vista, Controlador) para estructurar el sistema, además se han realizado otros diagramas como el de despliegue para indicar la relación con la base de datos y la plataforma Firebase además de detallar el diseño de la base de datos.

## Arquitectura

Se ha aplicado el patrón MVC como arquitectura de la aplicación, dividiendo por un lado las vistas con las que interactúa el usuario, los controladores que manejan las vistas y realizan operaciones de lógica de negocio y el modelo donde se guarda la información.

La vista está compuesta por diseños en ficheros en xml y se asocian a controladores siendo estos, actividades y fragmentos de Android incluyendo cuadros de diálogo. Los datos se almacenan como listas de elementos de tipo usuario, turnos y otros objetos del sistema.

Además se muestra la interacción con la base de datos Cloud Firestore y el servicio Firebase Authentication utilizando las API proporcionadas.

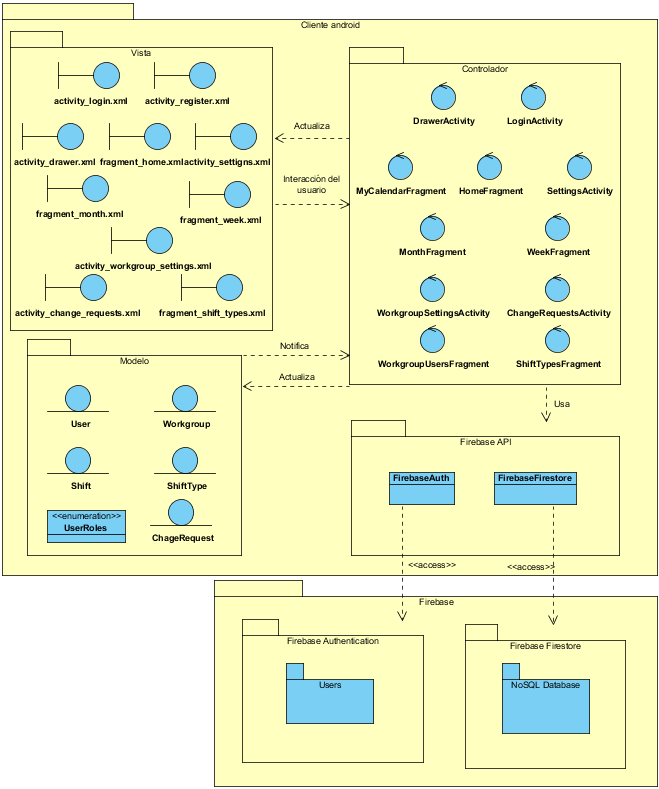


Ilustración - Diagrama de arquitectura MVC

## Diagrama de despliegue

En el modelo de despliegue se modelan los nodos físicos y se indica la composición del sistema en tiempo de ejecución indicando la comunicación con la plataforma Firebase para la gestión de usuarios y base de datos.

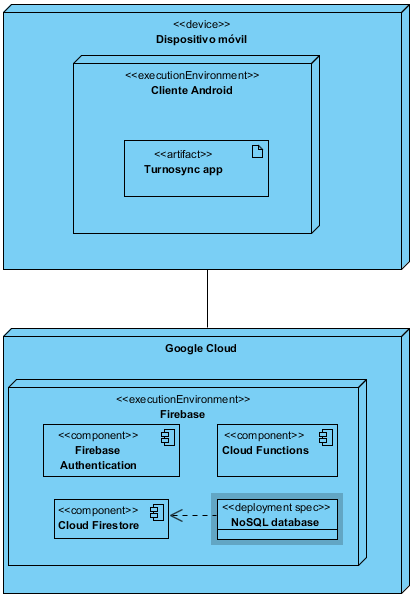


Ilustración - Diagrama de despliegue

## Diseño de la base de datos

La base de datos utilizada es una base de datos no relacional proporcionada por el servicio Cloud Firestore de Firebase. Se ha diseñado de tal forma que sea necesario el mínimo número de llamadas para conseguir la información requerida. La base de datos se compone de colecciones y documentos, siendo los documentos elementos que componen una colección y pueden contener subcolecciones.

Para el diseño se ha utilizado un diagrama de clases donde se indican los campos de cada documento.

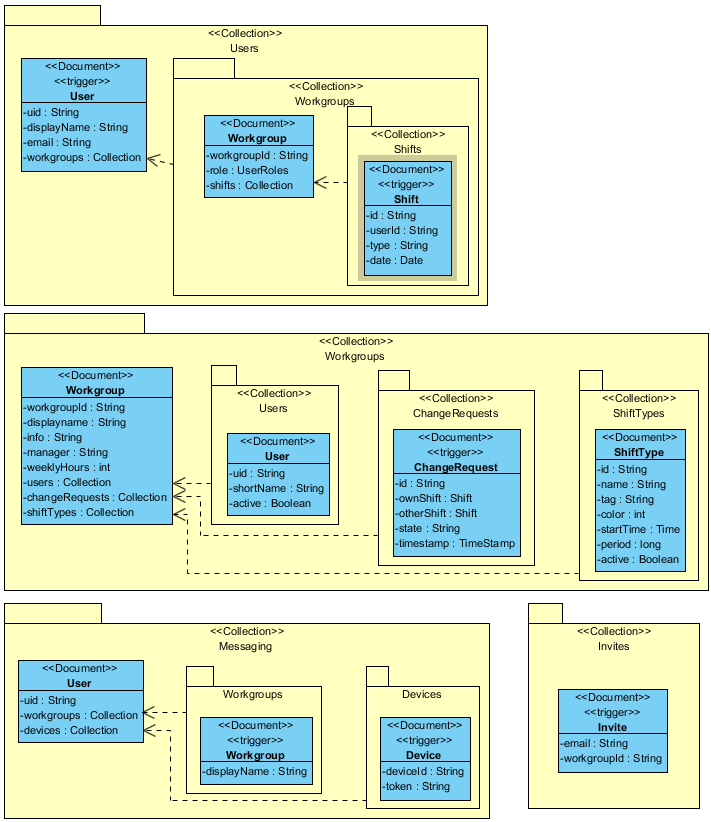


Ilustración - Diseño de base de datos

**Users**: Es una colección raíz que contiene los usuarios registrados en la aplicación. Estos usuarios tienen los siguientes campos:

* **Uid**: Identificador único del usuario obtenido del servicio de autenticación. Es de tipo String y es clave identificadora del documento.
* **DisplayName**: Nombre del usuario obtenido del servicio de autenticación. Es de tipo String.
* **Email**: Dirección de email del usuario obtenido del servicio de autenticación. Es de tipo String.
* **Workgroups**: Colección de referencias a los grupos de trabajo. Un elemento de esta colección tiene los siguientes campos:
  + **WorkgroupId**: Identificador de un grupo de trabajo, utilizado para obtener el resto de información. Es de tipo String y es clave identificadora del documento además de referencia de documentos Workgroup.
  + **Role**: Indica el rol del usuario dentro del grupo de trabajo, puede ser usuario o coordinador. Se usa una enumeración para definir este campo. Es de tipo String.
  + **Shifts**: Colección de turnos de trabajo del usuario dentro del grupo de trabajo. Cada turno tiene los siguientes campos:
    - **Id**: Identificador del turno de trabajo. Es de tipo String y es clave identificadora del documento.
    - **UserId**: Identificador del usuario asociado al turno. Es de tipo String y es referencia de documentos User.
    - **Type**: Identificador del tipo de turno que contiene la información. Es de tipo String y es referencia de documentos ShiftType.
    - **Date**: Fecha asociada al turno. Es de tipo Date.

**Workgroups**: Es una colección raíz que contiene los grupos de trabajo creados. Cada grupo de trabajo está compuesto por:

* **WorkgroupId**: Identificador de un grupo de trabajo, utilizado para obtener el resto de información. Es de tipo String y es clave identificadora del documento.
* **DisplayName**: Nombre del grupo de trabajo otorgado en su creación. Es de tipo String.
* **Info**: Campo de información extra sobre el grupo de trabajo.
* **Manager**: Identificador del usuario que actúa como coordinador, se establece a partir del usuario que crea
* **WeeklyHours**: Número de horas máximas establecidas para un usuario por semana. No limita directamente la asignación de turnos pero se usa para avisar al usuario de este hecho. Es de tipo Int.
* **Users**: Colección que contiene los usuarios que componen el grupo de trabajo. Utiliza referencias de los usuarios y contiene como campos:
  + **Uid**: Identificador único del usuario. Es de tipo String y es clave identificadora del documento además de referencia a documentos User.
  + **ShortName**: Nombre abreviado del usuario para poder mostrarlo en el calendario grupal. Se obtiene mediante las iniciales del nombre del usuario. Es de tipo String.
  + **Active**: Indicador de estado del usuario, activo por defecto, en caso contrario no se podrá asignar nuevos turnos al usuario ni aparecerá en la lista. Es de tipo Boolean.
* **ChangeRequests**: Colección de cambios de turnos dentro del grupo de trabajo. Cada cambio de turno tiene los siguientes campos:
  + **Id**: Identificador del cambio de turno. Es de tipo String y la clave identificadora del documento.
  + **OwnShift**: Turno del usuario que solicita el cambio. Es de tipo Object de la clase Shift.
  + **OtherShift**: Turno del usuario al que se le solicita el cambio. Es de tipo Object de la clase Shift.
  + **State**: Estado del cambio de turno que va cambiando en el proceso, se indica si está solicitado, aceptado, aprobado, rechazado o cancelado.
  + **TimeStamp**: Marca de tiempo en el que se solicita el turno. Es de tipo TimeStamp.
* **ShiftTypes**: Colección de tipos de turnos utilizados para definir turnos a los usuarios. Contiene la información general de los turnos y tiene como campos:
  + **Id**: Identificador del tipo de turno. Es de tipo String y la clave identificadora del documento.
  + **Name**: Nombre del tipo de turno. Es de tipo String.
  + **Tag**: Etiqueta del tipo de turno utilizada para representar los turnos de este tipo en el calendario. Es de tipo String.
  + **Color**: Color con el que representar los turnos de este tipo en el calendario. Es de tipo Int.
  + **StartTime**: Hora de comienzo del turno. Es de tipo Time.
  + **Period**: Duración del turno medido en milisegundos. Es de tipo Long.
  + **Active**: Indicador que indica si el tipo de turno es utilizable. Por defecto activo. Es de tipo Boolean.