DESARROLLO DE UN SISTEMA INFORMÁTICO PARA AYUDAR A ADULTOS CON SÍNDROME DE ASPERGER

Jefferson Almache Montoya
Juan Luis Armas Perona
María Salgado Iturrino

Grado en Ingeniería del Software
Facultad de Informática
Universidad Complutense de Madrid

Trabajo Fin de Grado en Ingeniería del Software

Dirigido por
Dra. Guadalupe Miñana Ropero

Madrid, junio 2016

Autorización de difusión

Madrid, a 17 de junio de 2016

Los abajo firmantes, alumnos y tutora del Trabajo Fin de Grado (TFG) en el Grado en Ingeniería del Software de la Facultad de Informática, autorizan a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores el TFG cuyos datos se detallan a continuación. Así mismo autorizan a la Universidad Complutense de Madrid a que sea depositado en acceso abierto en el repositorio institucional con el objeto de incrementar la difusión, uso e impacto del TFG en Internet y garantizar su preservación y acceso a largo plazo.

Desarrollo de un sistema informático para ayudar a adultos con síndrome de asperger

Curso académico 2015/2016

Firma de los alumnos

Firma del tutor

JEFFERSON ALMACHE MONTOYA

JUAN LUIS ARMAS PERONA

MARIA SALGADO ITURRINO

GUADALUPE MIÑANA ROPERO

Agradecimientos


```
switch(autor){
      case: Jeff
            agradecimientos(Juanlu);
            break;
      case: Juanlu
            agradecimientos(Jeff);
            break;
      case: María
            agradecimientos(Juan);
            break;
      default:
            A nuestras familias por su apoyo y cariño;
            A nuestra directora Guadalupe por su entusiasmo, dedicación e
            interés;
            A Victoria y a la Asociación Implica;
}
```

Resumen

Este proyecto presenta un sistema informático para ayudar a personas Asperger, que tienen problemas para recordar actividades y objetos básicos, y a sus profesores especialistas. Se ha decidido llamarlo AS (Asperger). Se compone de dos aplicaciones Android: la *aplicación tutor* y la *aplicación usuario.* La primera es para que los profesionales, desde su propia tablet, puedan centralizar y gestionar toda la información de sus alumnos creando tareas, retos y eventos específicos para cada uno. La segunda es para los móviles de las personas Asperger, se encarga de recordarles los sucesos que su tutor les ha asignado y después les pregunta si los realizaron correctamente. Ambas interfaces siguen los principios de claridad y sencillez, además la *aplicación usuario* es totalmente personalizable para que una persona Asperger de cualquier edad pueda interactuar y motivarse con ella.

La información es almacenada localmente en cada dispositivo. Ambas aplicaciones se comunican para sincronizar los datos mediante sockets usando la tecnología WIFI. Esto permite tener un seguimiento del progreso de cada alumno desde la *aplicación tutor.*

La implementación se ha realizado mediante una arquitectura multicapa que utiliza patrones de ingeniería del software para facilitar cualquier extensión o adaptación de la funcionalidad del sistema.

Abstract

This project presents a computer system designed to help Asperger people, who have trouble remembering basic objects and activities, and their specialist teachers. It is called AS (Asperger). It consists of two different Android applications: the tutor application and the user application. The first one is for professionals, it allows them to centralize and manage all theirs students information by creating tasks, events and challenges for each pupil from the teacher's mobile device. The second one is for Asperger students smartphones, it takes care to remind the activities that their tutor assigned to them and to ask if the student performed each activity correctly. Both interfaces follow the principle of clarity and simplicity, moreover the user application is fully customizable by any Asperger person of any age to ease the interaction and to motivate her.

The information is stored locally on each device. Both applications communicate to synchronize data via sockets using WiFi technology. This allows to track the progress of each student from the tutor application.

The implementation has used a multi-layer architecture using software engineering patterns to facilitate any extension or adaptation of the system functionality .

Palabras clave

ANDROID, SOCKETS, ASPERGER, NOTIFICACIONES, ALARMAS, ORMLITE, MULTICAPA, PATRONES, MONITORIZACIÓN Y APLICACIONES

Índice

Índice de figuras

CAPÍTULO 1: Introducción

"Sólo por diversión."

LINUS TORVALDS,

refiriéndose a las razones para haber creado Linux.

Este proyecto presenta un sistema informático, llamado *AS* (ASperger), que está diseñado para facilitar la interacción entre alumnos diagnosticados de Trastorno del Espectro del Autismo, concretamente síndrome de Asperger, y sus profesores especialistas. Esta enfermedad es un trastorno neuro-biológico severo del desarrollo en el cual existen desviaciones en diversos aspectos como las conexiones y habilidades sociales, la comunicación, la torpeza motora o características de comportamiento relacionados con rasgos repetitivos o perseverantes. Las personas Asperger necesitan directrices claras y muy explícitas para realizar cualquier actividad debido a su incapacidad para planificar y tomar decisiones. La labor de los profesionales que tratan con personas Asperger es bastante ardua ya que deben supervisar las rutinas de todos los que se encuentran bajo su responsabilidad diariamente. Hasta la fecha no existía ninguna herramienta tecnológica que apoyará esta labor.

AS es un sistema de monitorización que surge para cubrir esta necesidad con dos objetivos fundamentales. Por una parte, simplificar el quehacer de los profesores especialistas facilitándoles un seguimiento de sus alumnos Asperger mediante el control constante de las tareas que realizan a través de un dispositivo electrónico (tablet). Y, por otra parte, ayudar a estas personas, mediante recordatorios y notificaciones, a realizar sus tareas diarias mejorando su calidad de vida.

Otro aspecto importante que motiva la realización del proyecto es que a día de hoy existen pocas aplicaciones centradas en estas personas. La mayoría solo se encuentran dirigidas a niños o se basan en otras enfermedades más comunes, dejando de lado a adolescentes o adultos que padecen este trastorno. Como dice Victoria, gerente de Asociación Implica: "Los niños con autismo también crecen y pasan a ser adolescentes, es muy importante que nos demos cuenta de esto porque hay que enseñarles a tratar con adolescentes normales. Esto es lo bueno de *AS*."

El trabajo se ha desarrollado en un contexto multidisciplinar donde han colaborado, con nosotros y nuestra directora, miembros de la Asociación Implica proporcionando la información necesaria para desarrollar *AS* y permitiendo la realización de pruebas.

Como se puede ver en la fig. 1, el sistema que hemos desarrollado está compuesto por dos aplicaciones móviles distintas, ejecutables en dispositivos Android (smartphones y tablets). La *aplicación tutor* está destinada a los profesores y la *aplicación usuario* está dirigida a las personas Asperger. Ambas se conectan y sincronizan intercambiando información mediante sockets. Al no haber necesidad de servidores intermedios, ya que cada aplicación tiene una base de datos local, se reducen los costes de mantenimiento de la aplicación y además se eliminan bastantes posibles problemas relacionados con la privacidad y protección de datos. La funcionalidad y detalle se explicará en más profundidad en los capítulos 6, 7 y 8 de este documento.

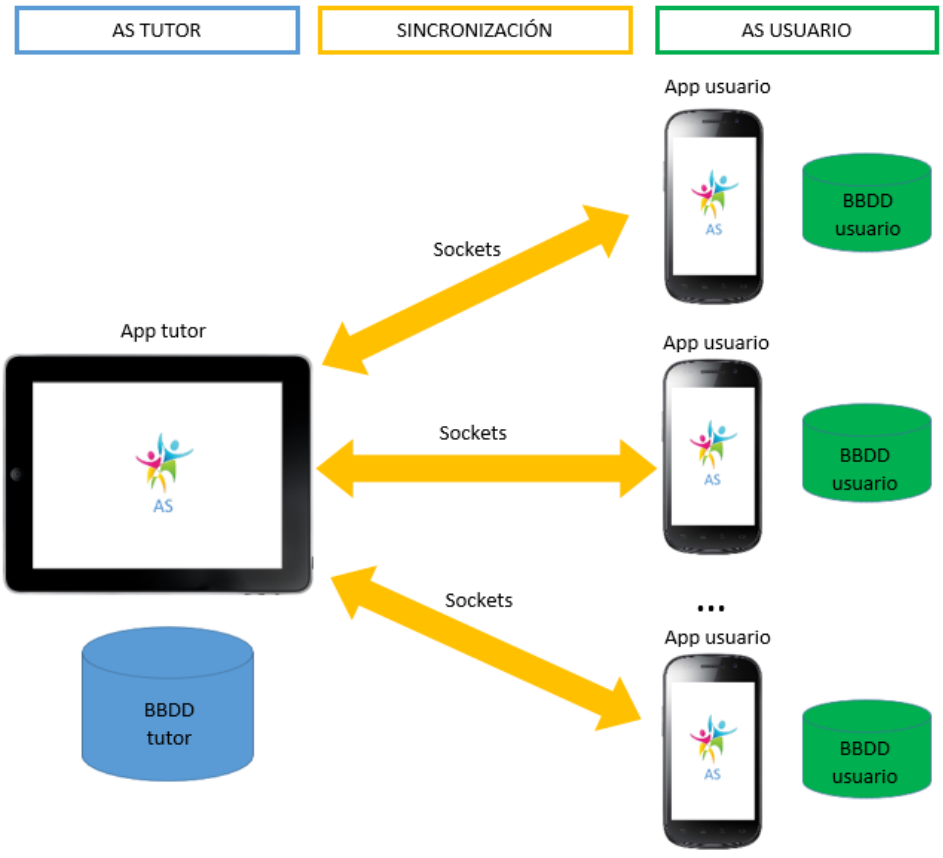


Figura 1: Diagrama de la arquitectura global del sistema

El resultado de este proyecto ha sido una herramienta integral y fiable que, permite a un profesor tener centralizada toda la información de un número ilimitado de alumnos Asperger y poder gestionarla de una manera intuitiva y amigable. El profesor puede crear, desde su

dispositivo usando la *aplicación tutor*: tareas para el día a día, eventos puntuales o retos para cada alumno. Todos los cambios se actualizan cuando se realiza una sincronización con la *aplicación usuario* instalada en el móvil de la persona Asperger. Esta aplicación se encarga de recordar sus tareas y eventos directamente al alumno con la frecuencia personalizada por el profesor y después del tiempo que corresponda se le hace una pregunta para comprobar si el usuario realizó la actividad correctamente. Las respuestas se almacenan, se procesan y se envían a la *aplicación tutor* cuando se realiza una sincronización para facilitar el seguimiento de la evolución del alumno. Además, la *aplicación usuario* cuenta con un proceso automático que en base al progreso de cada tarea juega con las frecuencias de las alarmas. Si se detecta que el usuario ha interiorizado una tarea debido a que sus respuestas son siempre afirmativas se pasa a recordarle esa actividad con menor frecuencia. Por el contrario, si se detectan olvidos en realizar esa actividad se vuelve a frecuencias más insistentes.

En las siguientes figuras se muestra algunas de las funcionalidades de la *aplicación tutor.* En la fig. 2 aparece una vista con el listado de usuarios dados de alta en el sistema y el detalle del usuario seleccionado. En la fig. 3 se muestra la vista correspondiente a la opción de crear un evento, en ella se puede seleccionar cómodamente a que alumnos del sistema invitar y, una vez creado, mostrará quiénes asistirán.

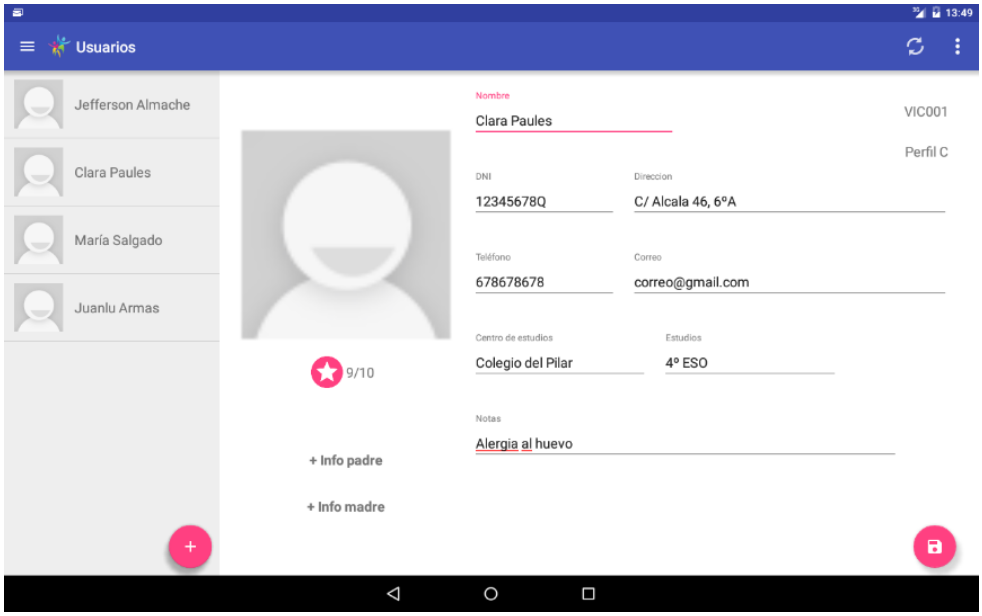


Figura 2: Listado de usuarios dados de alta en la *aplicación tutor*

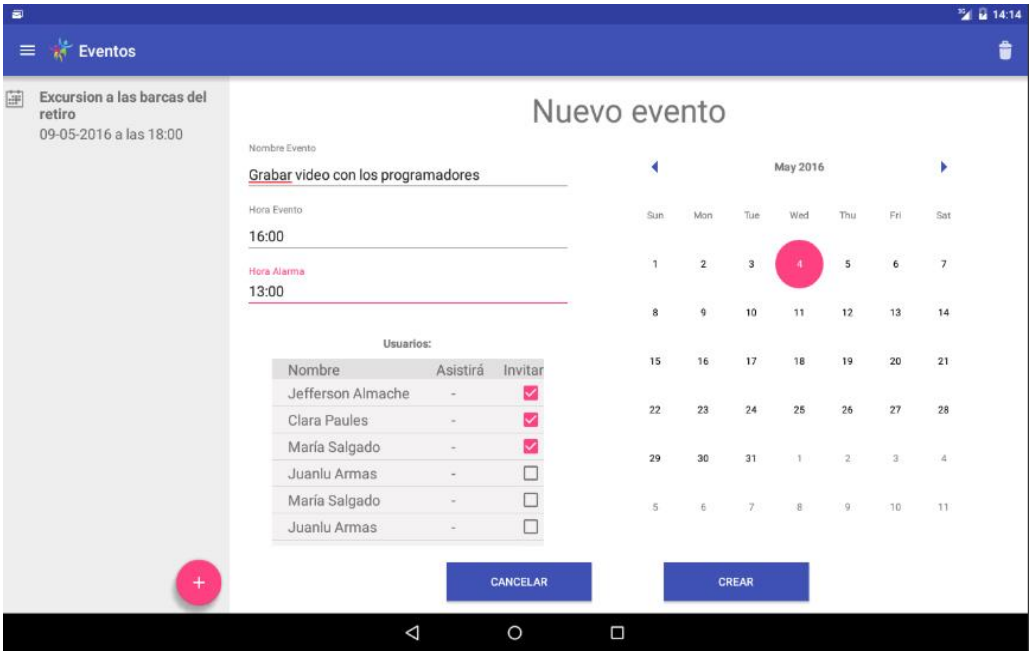


Figura 3: Creación de un evento en la *aplicación tutor*

La fig. 4 muestra algunas de las funcionalidades de la *aplicación usuario.* La imagen de la izquierda muestra la pantalla principal con las posibles opciones que ofrece la aplicación. La imagen del centro muestra la vista asociada a un reto, se puede ver el progreso hasta llegar a la meta y el premio que se le propone como aliciente. La imagen de la derecha muestra la lista de eventos a los que ha sido invitado en donde puede indicar si asistirá o no.

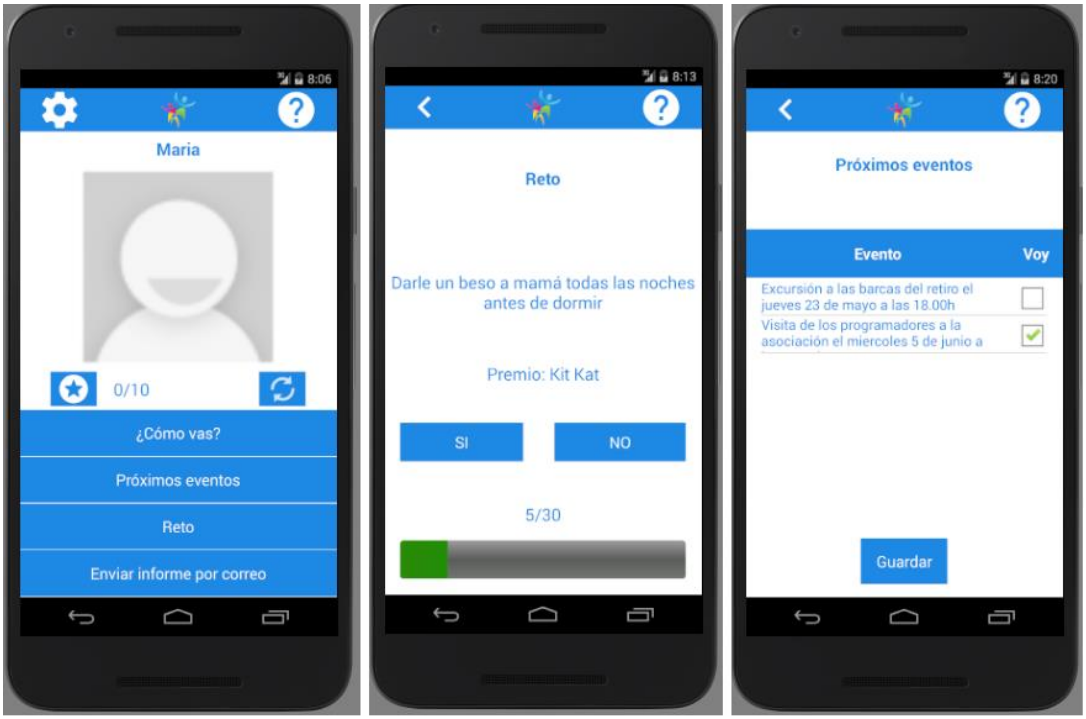


Figura 4: Algunas funcionalidades de la *aplicación usuario*

AS es un sistema gratuito porque durante el desarrollo de este proyecto no buscamos lucrar sino combinar los fines académico-educativos propios de un trabajo de final de grado con un fin solidario y altruista como es mejorar la calidad de vida de personas que padecen autismo.

En esta primera versión, *AS* se reduce al ámbito académico y al perfil de personas Asperger pero todo el sistema ha sido implementado de manera modular utilizando patrones de ingeniería del software para hacer posible la adaptación a otros entornos como asociaciones o residencias y a otras enfermedades, por ejemplo el Alzheimer.

Estamos muy contentos porque en el impulso por crear algo que contribuya a la sociedad, hemos usado los conocimientos y técnicas adquiridas en el grado. Además, este proyecto nos ha obligado a aprender tecnologías completamente nuevas para nosotros, principalmente el desarrollo Android. También nos ha permitido relacionarnos con clientes lo cual ha supuesto una dificultad añadida debido a continuos cambios en los requisitos y a otros problemas que aparecen cuando se desarrolla e implementa un sistema informático real. Todo esto nos hace darnos cuenta de que las asignaturas que hemos estudiado en el grado a lo largo de estos años nos han dado una base de conocimientos suficientemente amplia para poder llevar a cabo satisfactoriamente un proyecto de esta envergadura.

Esta memoria está organizada de la siguiente manera:
El presente capítulo introductorio donde se encuentra una breve explicación del proyecto, la motivación que nos llevó a desarrollarlo y sus aspectos fundamentales. En el capítulo 2 se repite la introducción en inglés. El capítulo 3 presenta el estado del arte en el que se habla de las distintas soluciones tecnológicas relacionadas con el autismo y el Asperger existentes hasta el momento y otras aplicaciones en las que nos hemos basado. En el capítulo 4 se ha realizado un breve repaso de las tecnologías utilizadas en este proyecto. En el capítulo 5 se explica la arquitectura del sistema con los distintos patrones de ingeniería del software que se han utilizado en el proyecto. En los capítulos 6 y 7 se explican las dos aplicaciones Android implementadas, la *aplicación usuario* y la *aplicación tutor*. En el capítulo 8 se explica cómo se comunican los dos tipos de aplicación. Tras explicar todo el diseño del sistema, en el capítulo 9 se detallan los casos de uso. En los capítulos 10 y 11 se recogen los resultados obtenidos después del período de desarrollo y de pruebas y se presentan las conclusiones y el trabajo futuro en español e inglés. En el capítulo 12 se detalla el trabajo de cada uno de los integrantes del equipo que hemos realizado este este proyecto. Al final se puede consultar al final la bibliografía y las referencias. Y, por último, en el anexo A se muestran los diagramas de clase y en el anexo B el cronograma de la planificación.

CHAPTER 2: Introduction

This project is an informatic system called AS (ASperger). It is designed to ease the interaction between people diagnosed with autism spectrum disorder, specifically Asperger's syndrome, and theirs specialist teachers. This illness is a severe neuro-biological developmental disorder in which there are deviations in various aspects such as connections and social skills, communication, clumsiness or behavioral characteristics associated with recurrent or persistent features. Asperger people need clear and very explicit guidelines for any activity due to their inability to plan and make decisions. The work of professionals who deal with Asperger people is pretty hard because they must monitor daily all the routines of each pupil who is under their responsibility. To date there was no technological tool to support this work.

AS is a monitoring system that arises to fill this need with two fundamental objectives. On one hand, to simplify the work of specialist teachers, providing them with a tracking of theirs Asperger students through a constant control of the tasks they perform. On the other hand, to help Asperger people through reminders and notifications to perform their daily tasks improving their life quality.

Another important aspect that motivates the project is that today, there are just a few applications focused on these people. Most are only aimed at children or based on other more common diseases, leaving aside teenagers or adults who suffer from this disorder. As Victoria, manager of Implica Association, says: "Children with autism also grow up and become teenagers. It is very important that we realize this because you have to teach them to deal with normal adolescents. This is a great thing of AS".

The project has been developed in a multidisciplinary context having collaborated with us and our directress, members of Implica Association who have provided us the necessary information to develop AS and have allowed us to test it .

As you can see in fig. 5, the system we have developed is composed of two different mobile applications, executable on Android devices (smartphones and tablets): the *tutor application* which is for teachers and the *user application* which is for Asperger students. Both connect and synchronize exchanging information via sockets. With no need of intermediate servers, as each application has a local database, maintenance costs are reduced and also some possible

problems related to privacy and data protection are removed. Functionality and detail will be explained in more depth in Chapters 6 and 7 of this document.

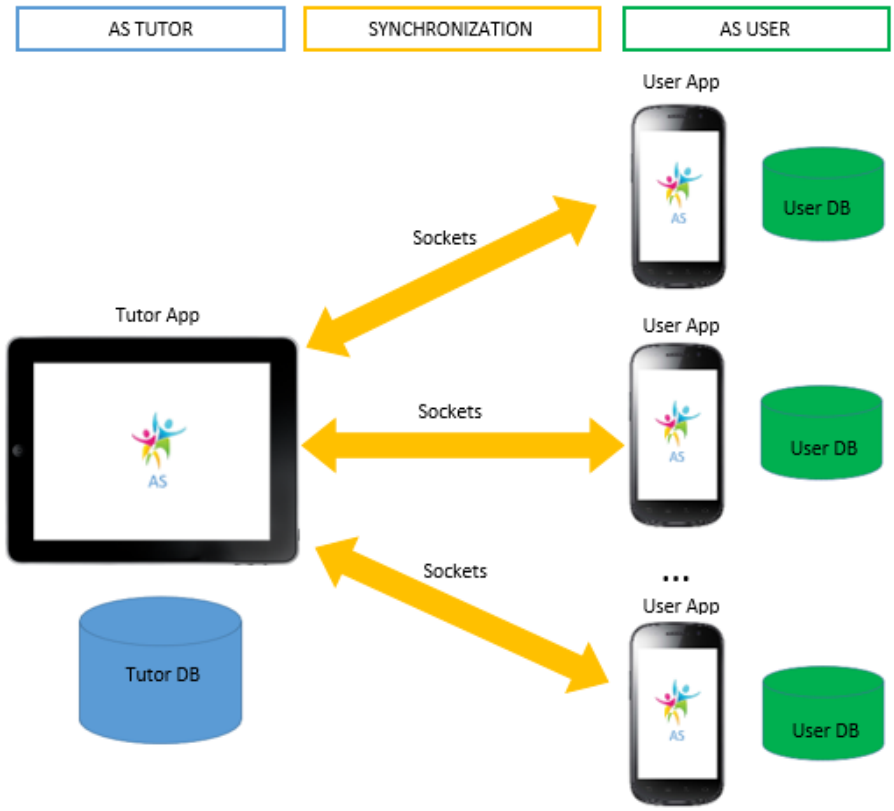


Figura 5: Global architecture system diagram

The result of this project is an integral and reliable tool that allows a teacher to centralize all the information of an unlimited number of Asperger students and able to manage in an intuitive and friendly way. The teacher can create, from his device using the *tutor application*: tasks for the day, specific events or challenges for each student. All changes are updated when a synchronization with the *user application*, installed on the mobile of the Asperger person, is done. This application remembers tasks and events directly to the student with a personalized frequence by the teacher and after the appropriate time it asks a question to check if the user has performed the activity correctly. The answers are stored, processed and sent to the *tutor application* when a synchronization is done to facilitate the monitoring of the student evolution. In addition, the *user application* has an automated process based on the progress of each task which plays with the frequency of the alarms. If it is detected that the user has internalized a task because its answers are always positive, the reminders will have less frequence. Conversely, if oversights are detected in that activity, it comes back to more insistent frequencies.

The following figures show some of the features of the *tutor application*. Fig. 6 displays a view with the list of users registered in the system and the detail of the selected user. Fig. 7 shows the view corresponding to the option of creating an event. On this screen you can comfortably select the students you want to invite and, once created, it will show who will attend.

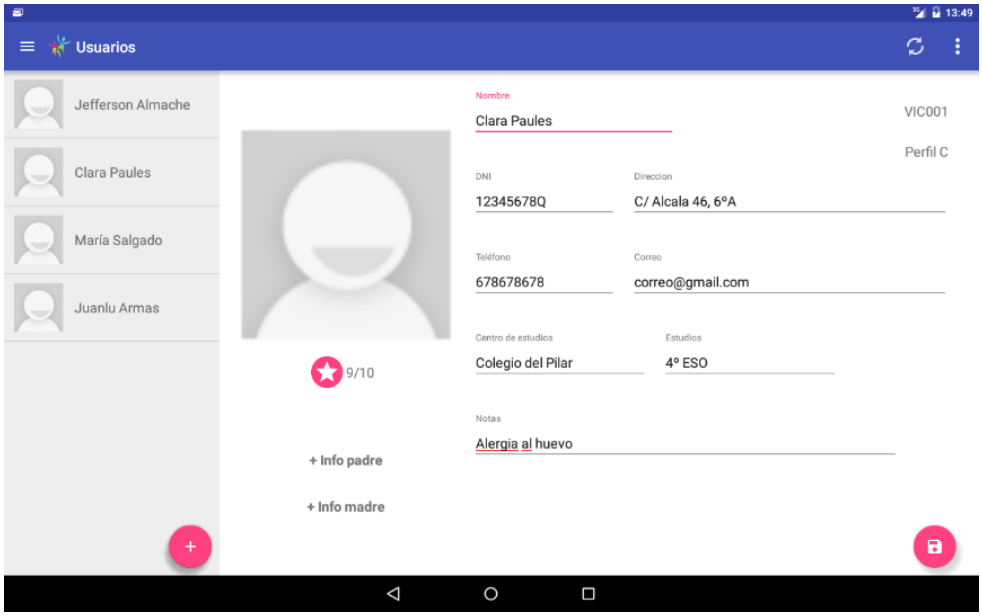


Figura 6: List of registered users in the *tutor application*

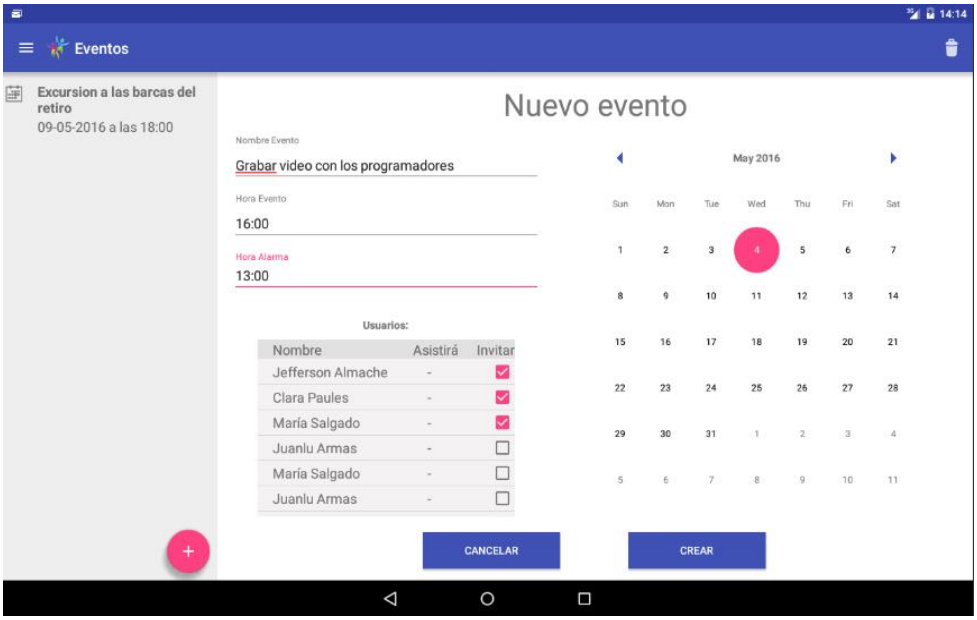


Figura 7: New event creation screen in the *tutor application*

Fig. 8 shows some of the features of the *user application*. The image on the left shows the main screen with the options offered by the application . The center image shows the view associated with a challenge where you can see the progress to reach the goal and the prize proposed as an

incentive. The image on the right shows the list of events you have been invited, where you can indicate whether or not you will attend.

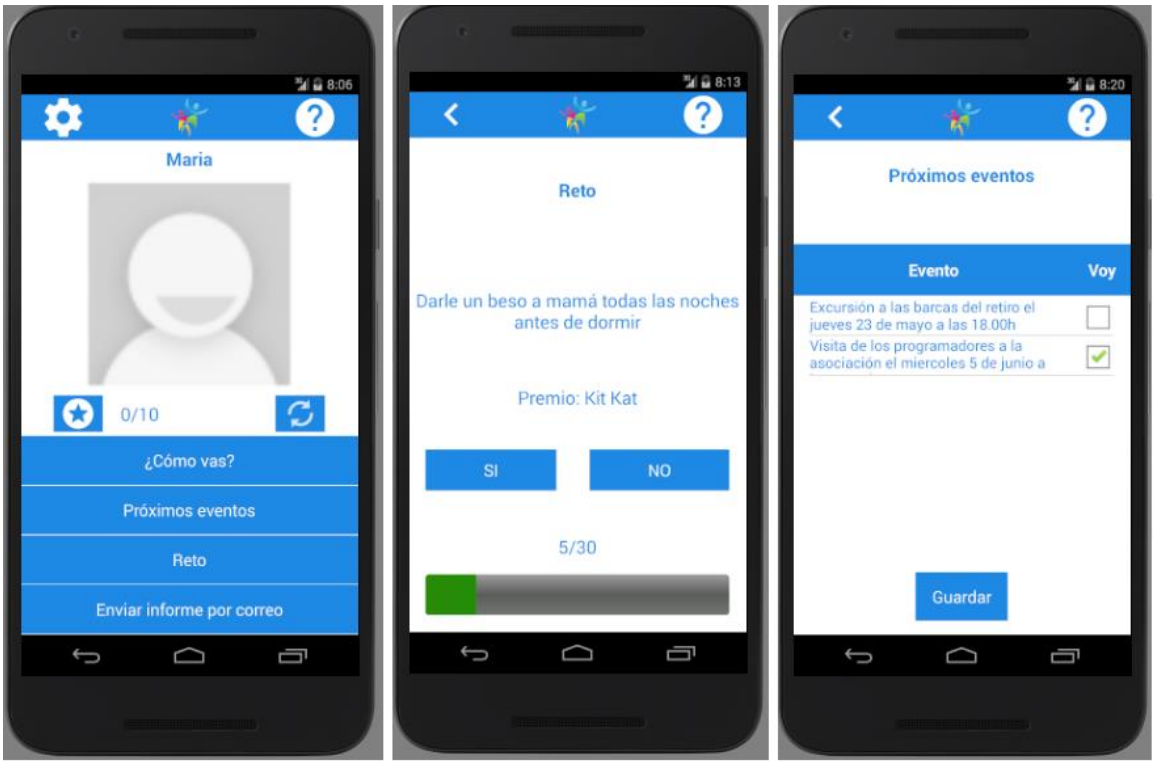


Figura 8: Some features of the *user application*

AS is a free system because during the development of this project we do not seek profit but combine the academic and educational purposes of a final degree work with a supportive and altruistic aim, like improving the life quality of people with autism.

In this first version, AS is reduced to the academical field and to the Asperger scope but the entire system has been implemented in a modular way by the use of software engineering patterns to enable the adaptation to other environments such as associations or residences and other diseases, such as Alzheimer.

We are very happy because in the drive to create something that contributes to society , we have used the knowledge and skills acquired in the degree . Furthermore, this project has forced us to learn new technologies for us, mainly Android development. It has also allowed us to relate with actual customers which meant an added difficulty due to the continuous changes in requirements and other problems found when developing and implementing a real computing system. All this, makes us realize that the subjects we have studied in the degree over the years have given us enough base knowledge to successfully carry out a project of this magnitude.

This document is organized as follows:

This introductory chapter is a brief explanation of the project, the motivation that has led us to develop and fundamental aspects. In chapter 1 the introduction is in Spanish and in the present chapter it is in English. Chapter 3 presents the state of the art in which we talk about the different technological solutions related to autism and Asperger existing so far and other applications we have relied on. In chapter 4 there is a brief overview of the technologies used in this project. Chapter 5 explains the architecture of the system with the different software engineering patterns that have been used in the project. In chapters 6 and 7 the two deployed Android applications are explained: the *user application* and the *tutor application*. Chapter 8 explains how the two types of application communicate between them. After explaining the whole design of the system, in chapter 9 all the use cases are detailed. Chapters 10 and 11 collect the results after the period of development and testing and present the conclusions and future work in Spanish and English. In chapter 12 the work of each of the team members who have made this project is detailed. Bibliography and references can be found at the end. Finally, annex A shows the class diagrams and annex B shows the schedule planning .

CAPÍTULO 3: Estado del arte

"La imaginación es más importante que el conocimiento.
El conocimiento es limitado, mientras que la imaginación no."
ALBERT EINSTEIN

La tecnología es una gran aliada para ayudar en el desarrollo de personas con trastornos como el autismo. Durante los últimos años varias fundaciones han promovido la creación de soluciones tecnológicas que ayuden a estas personas. Víctor Rodríguez, director clínico de "Planeta Imaginario" recuerda que hace ocho años le decían "que las nuevas tecnologías, y sobre todo el mundo de las tablets iban a ser el futuro, y nos habían hablado de que en el tratamiento de niños con necesidades especiales podrían ser una herramienta diferenciadora. En ese momento creímos que aquella gente estaba loca… Y mira ahora. ¡Eran visionarios!". Por estos motivos, existen multitud de aplicaciones móviles que permiten mejorar a estas personas en habilidades concretas. A continuación se recogen algunos ejemplos. [1, 2]

3.1 Aplicaciones para personas con autismo

- ***Grace***

  Los autistas se caracterizan por tener dificultades para comunicarse con los demás. *Grace* es una aplicación que ayuda a estas personas a comunicar sus necesidades de forma autónoma mediante un **sistema de intercambio de imágenes sin audio**. El objetivo es formar frases con significado a través de la selección de imágenes que además sirven para aprender y ayudarles a pronunciar las palabras.

Figura 9: Logo aplicación *Grace*

➢ ***Training Faces***

Esta aplicación se enfoca en aprender a **reconocer las emociones a través de las expresiones faciales**. *Training Faces* ayuda al autista a ser consciente de por qué alguien sonríe cuando está contento o llora cuando está triste, lo que mejora las habilidades sociales mediante fotografías de caras reales como se muestra en la fig. 10.

Figura 10: Captura de la aplicación *Training Face*s

➢ ***Niki Story***

La falta de retención y de memoria son también problemas que genera el autismo. *Niki Story* tiene el objetivo de estimular la **habilidad narrativa, la comprensión y el pensamiento**, fomentando la autonomía y la imaginación.

La app consiste en crear historias multimedia, álbumes de fotos, cuentos hablados y tareas a través de imágenes, textos con símbolos, grabación de voz, vídeo e incluso dibujos hechos directamente en la pantalla con el dedo. Además, permite importar y exportar historias y leerlas en PDF.

Figura 11: Logo de la aplicación *Niki Story*

➢ ***Abilipad***

Se trata de un **editor de texto adaptable y personalizable** con voz y predicción de palabras para niños con problemas especiales. *Abilipad* permite crear fácilmente diseños de teclados personalizados, pudiendo asignar a cada tecla una letra, palabra, frase o dibujo, grabaciones de audio y seleccionar varias tipografías y tamaños de letra. Se puede ver su llamativa interfaz en la fig. 12.

Figura 12: Captura de una pantalla de la aplicación *Abilipad*

➢ ***Slide & Spin***

Esta app está diseñada para ayudar a niños a partir de un año y medio a desarrollar **habilidades de motricidad**. El niño aprenderá los cuatros movimientos básicos de dar vuelta, deslizar, girar y presionar un botón a través de un juego educativo que esconde sorpresas debajo de cada botón como se muestra en la fig. 13. A través de colores, imágenes y voz, esta app motiva al usuario a desarrollar sus habilidades y su interés por las cosas que le rodean.

Figura 13: Captura de una pantalla de la aplicación *Slide & Spin*

➢ ***VAST Autismo 1***

Vast Autismo tiene el objetivo de ayudar con las **habilidades de vocalizar, reconocer y usar el lenguaje hablado** con mayor claridad y calidad. Se basa en una serie de vídeos y audios que centrados en una de las actividades que el usuario tiene que repetir hasta que la aprende tal y como se muestra en la fig. 14. Este método está organizado en seis niveles que aumentan la dificultad: repetición de sílabas, sinónimos, artículos, verbos, frases y oraciones.

Figura 14: Fotograma de un video de la aplicación *VAST Autismo 1*

➢ ***Sígueme***

Sígueme es una herramienta educativa desarrollada por la Fundación Orange y la Universidad de Granada para **potenciar el desarrollo de procesos perceptivo-visual y cognitivo-visual**. La app se presenta en 6 fases que van desde la estimulación basal a la adquisición de significado a partir de vídeos, fotografías, dibujos, pictogramas y juegos.

Algunos de los objetivos de la herramienta son: mejorar la atención visual, desarrollar la capacidad de observación, potenciar la representación mental y la comprensión lingüística, etc.

Figura 15: Splash screen de la aplicación *Sígueme*

3.2 Aplicaciones para personas con Asperger

Existen bastantes aplicaciones enfocadas específicamente a enfermos de Asperger[3]. Una de ellas, ***Aba Planet***, se encuentra en el ranking de las 50 mejores apps de salud en español. Esta aplicación permite a grandes y pequeños trabajar con las formas de los objetos y el vocabulario más básico, clasificado en 18 categorías: vehículos, ropa, comida... La app se adapta al niño, decidiendo los contenidos para trabajar según su ritmo de aprendizaje. Sirve tanto para trabajar en casa como en terapia.

Figura 16: Captura de la pantalla principal de *Aba Planet*

Otra aplicación en este campo es ***iSecuencias*.** Se basa en 100 escenas preprogramadas que permiten múltiples combinaciones de ejercicios con los que aprender estructuras básicas del lenguaje o percibir las emociones. Está especialmente diseñada para niños con síndrome de Asperger o autismo de alto funcionamiento.

Figura 17: Captura de una pantalla de la aplicación *iSecuencias*

3.3 Aplicaciones de progreso automático

Uno de los objetivos de AS es que el profesor pueda despreocuparse de tener que recordar a sus alumnos sus tareas rutinarias, pero también se quiere fomentar la autonomía de las personas Asperger para no hacerlas totalmente dependientes de la aplicación. Por esto hemos pensado un sistema automático que en base al progreso del alumno para cada tarea juega con las frecuencias de las alarmas. Si se detecta que el usuario ha interiorizado una tarea debido a que sus respuestas son siempre afirmativas se pasa a recordarle esa actividad con menor frecuencia. Por el contrario, si se detectan olvidos en realizar esa actividad se vuelve a frecuencias más insistentes.

Para diseñar este proceso automático nos hemos basado en la idea de ***Duolingo*** ya que esta aplicación, que sirve para practicar idiomas, trabaja más aquellos aspectos en los que el usuario comete fallos y edita la frecuencia con la que trabaja el vocabulario en base al número de aciertos. Se van desbloqueando nuevos niveles pero nunca se descuidan los niveles ya completados.

Figura 18: Logo de la aplicación *Duolingo*

3.4 Aportaciones de nuestra aplicación

AS es una aplicación distinta a todas las que existen actualmente por múltiples razones:

- ➢ Encontramos muchísimo material para el desarrollo intelectual y de las capacidades de comunicación, pero abordando el tema de la planificación todavía no se ha desarrollado ninguna aplicación que permita a un profesor especialista o tutor gestionar la planificación de una persona enferma desde su propio dispositivo sincronizando la información con el dispositivo del alumno.
- ➢ Tampoco existe ninguna aplicación orientada a la monitorización de un grupo de personas con trastornos desde un solo tutor. Esto cobra especial interés para monitores de grupo o encargados de asociaciones.
- ➢ Las aplicaciones que se encuentran a disposición de los usuarios están orientadas a un público infantil. Esto puede causar vergüenza o no satisfacer las necesidades de un usuario adulto o adolescente. Por este motivo AS dispone de un modo por defecto de carácter formal y discreto, aunque incluye varias posibilidades de personalización.
- ➢ La mayoría de las aplicaciones para un perfil tan concreto como el autismo o el asperger son de pago. Haciendo un sondeo sobre las aplicaciones disponibles en un buscador de aplicaciones para autistas la mayoría son de pago y un 60% supera los 50€. AS es diferente porque es gratuita.

CAPÍTULO 4: Tecnologías utilizadas

"En la confrontación entre el arroyo y la roca,
el arroyo siempre ganará,
no por la fuerza,
sino por la persistencia."
BUDDHA

En este capítulo se describen las herramientas tecnológicas que se han utilizado en cada una de las fases del desarrollo de AS.

4.1 Documentación y comunicación

- ***Google Drive***

Figura 19: Logo de Google Drive

Google Drive es un servicio de alojamiento de archivos que permite la creación y modificación de documentos en línea con la posibilidad de colaborar en grupo. Es una herramienta muy cómoda para empezar a planificar y escribir datos de nuestro proyecto. No requiere un aprendizaje previo como programas de edición de LaTeX o Microsoft Project. Hemos usado este servicio a lo largo de nuestro trabajo fin de grado para organizarnos y almacenar los documentos e imágenes que necesitábamos para seguir implementando nuestra aplicación. No hemos almacenado código aquí. También hemos utilizado diversas extensiones de Google Docs como: *Table of contents*, que genera una barra lateral, con un índice de las secciones de un documento, así se puede navegar más cómodamente por él; *Captionizer*, una herramienta que favorece la creación de los pies de foto de las imágenes de un texto así como la creación de un índice para estas; y *ProQuest RefWorks* para gestionar la bibliografía.

4.2 Diseño

- ***IBM Rational Software Architect 8.0***

Figura 20: Logo aplicación "IBM RSA"

IBM RSA es una avanzada y exhaustiva aplicación de diseño, modelado y herramienta de desarrollo software. Emplea el lenguaje de modelado unificado (UML por sus siglas en inglés) para diseñar arquitecturas en C++ o Java. Incorpora una infraestructura de software de código abierto Eclipse. Hemos usado esta aplicación para realizar los diagramas de secuencia y actividad de nuestro proyecto.

- ***Pencil***

Figura 21: Logo herramienta "Pencil"

Pencil es una herramienta gratuita y open-source para diseñar interfaces gráficas. Es fácil de instalar y usar. Incluye varias plantillas para crear diseños en diferentes plataformas conocidas. Hemos usado esta herramienta en la fase de diseño de la aplicación para realizar nuestros primeros bocetos, imaginando cómo serían las pantallas de cara a las primeras reuniones con el cliente y para hacer más visual nuestra idea de la aplicación.

4.3 Desarrollo

- ***Android Studio***

Figura 22: Logo Android Studio

Android Studio es un entorno de desarrollo integrado claro y robusto para la plataforma Android. Está disponible para los sistemas operativos Windows, Mac OS X y Linux. Contiene un amplio repertorio de herramientas para ayudar en el desarrollo de las aplicaciones, así como la posibilidad de probar la aplicación en diferentes dispositivos virtuales y en diferentes versiones de Android. Hemos usado este entorno para desarrollar la gran parte del código de nuestro proyecto.

- ***Gradle***

Figura 23: Logo herramienta construcción "Gradle"

Gradle es una herramienta para automatizar la construcción de proyectos, es decir la compilación o el testing por ejemplo, además gestiona las dependencias. Está basada en Groovy, un lenguaje de programación muy parecido a Java y está incorporada en Android Studio.

4.4 Control de versiones

- ***Git***

Figura 24: Logo software control versiones “Git”

Git es un software de control de versiones gratuito y open source. Está disponible para las plataformas más conocidas como son Mac OS X, Windows, Linux y Solaris. Requiere aprender una serie de conceptos para poder usarlo correctamente. Hemos usado este software para gestionar nuestro repositorio de Github desde nuestros ordenadores personales.

- ***Github***

Figura 25: Logo plataforma “GitHub”

GitHub es una plataforma de desarrollo colaborativo para alojar software utilizando el control de versiones Git. En las cuentas gratuitas el código se guarda de forma pública sin embargo en las cuentas de pago se puede almacenar de forma privada. Usamos esta plataforma para gestionar y facilitar el desarrollo del código de nuestra aplicación.

4.5 Base de datos

- ***SQLite***

Figura 26: Logo motor de bases de datos “SQLite”

SQLite es un motor de bases de datos de código abierto, transaccional, que no necesita configuración ni requiere un servidor y se caracteriza por mantener el almacenamiento de información de forma sencilla. Es una tecnología cómoda para los dispositivos móviles a pesar de que tiene ciertas limitaciones en determinadas operaciones. Hemos usado este motor para crear y testear la base de datos que vamos a usar en la aplicación para Android.

- ***ORMLite***

Figura 27: Logo libreria “OrmLite”

ORMLite (Object Relational Mapping Lite) es una librería open source que permite mapear objetos Java y persistirlos en una base de datos SQL. Utiliza clases abstractas para el acceso a los datos (DAO) y soporta SQLite. Dado que nuestro proyecto usamos esa base de datos hemos utilizado esta librería para gestionar las entidades así como los datos de las aplicaciones. Como estudiantes del Grado de Ingeniería del Software queríamos emplear el máximo de nuestros conocimientos previos, por eso decidimos persistir nuestros datos a través de JPA, sin embargo debimos desechar esta idea dado que es un API no adecuada para Android, entonces decidimos usar ORMLite que si soporta SQLite y no nos generaba ningún problema.

➢ ***DB Browser for SQLite***

Figura 28: Logo gestor de bases de datos “DB Browser for SQLite”

DB Browser for SQLite es una herramienta open source con la que gestionar archivos de bases de datos SQLite disponible para Windows, Mac OS X, Linux y FreeBSD. Nos ha permitido inspeccionar el contenido de las bases de datos que generaba nuestra aplicación, dentro de los emuladores. Cuando se importan y se abren con esta herramienta, aparecen claramente las distintas tablas con sus columnas, lo cual permite comprobar el correcto o incorrecto manejo de los datos .

CAPÍTULO 5: Arquitectura del sistema AS

"Programar sin una arquitectura o diseño en mente
es como explorar una gruta sólo con una linterna:
no sabes dónde estás, dónde has estado ni hacia dónde vas."
DANNY THORPE

En este capítulo se explica la arquitectura y los patrones de ingeniería de software que se han utilizado en el sistema AS. Para explicar en detalle cómo es el funcionamiento de la arquitectura conviene recordar su organización global(ver figura 29).

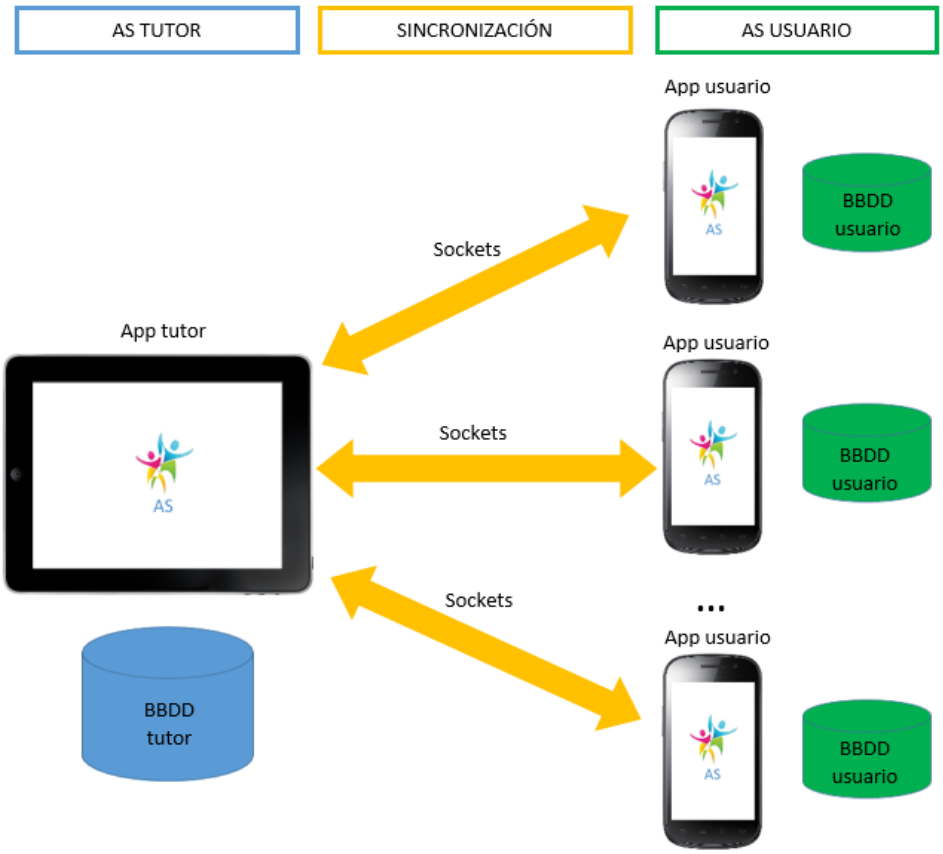


Figura 29: Diagrama de la arquitectura global del sistema

5.1 Arquitectura multicapa

Para desarrollar las aplicaciones tutor y usuario se ha decidido utilizar la arquitectura multicapa ya que nos proporciona integración y reusabilidad, encapsulación, distribución, escalabilidad, manejabilidad, mejora el rendimiento y mejora la fiabilidad.

Como se puede ver en la figura 30 la arquitectura multicapa se divide en tres capas: presentación, negocio e integración o acceso a los datos. La capa de presentación es la encargada de la interacción directa con el usuario, es la responsable de proporcionar los servicios del sistema, además debe ser amigable, entendible y fácil de usar. La capa de negocio proporciona las reglas y servicios del sistema, es la encargada de recibir las peticiones que le llegan de la capa de presentación, comunicarse con la capa de integración, procesar los datos y devolver los resultados. La capa de integración es responsable de la comunicación con recursos y sistemas externos, normalmente está formada por uno o más gestores de bases de datos.

Figura 30: Esquema arquitectura multicapa

En el sistema que hemos desarrollado, la arquitectura multicapa utiliza patrones de Ingeniería de Software para conseguir la separación de las capas de presentación, negocio e integración. Los patrones empleados son: singleton, dispatcher, command, factoría abstracta, servicio de aplicación, DAO y data transfer object (DTO). Todos estos patrones serán explicados en el subapartado 5.1. La arquitectura multicapa del sistema AS a grandes rasgos sería como se muestra en la figura 31.

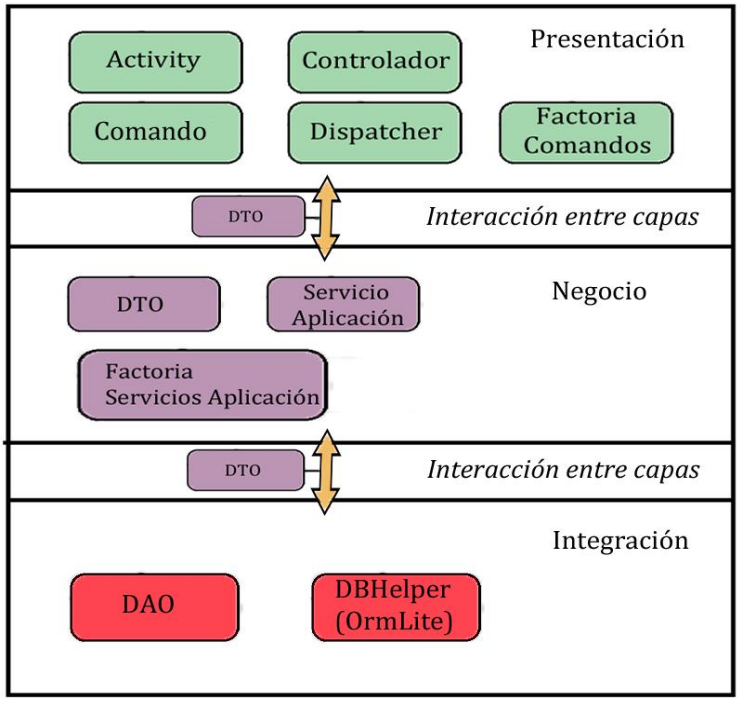


Figura 31: Arquitectura multicapa AS

Para explicar el flujo del sistema y su funcionamiento vamos a poner un ejemplo desde la *aplicación tutor*: crear reto("Lavarse las manos después de comer"). En la capa de presentación el usuario de la aplicación hace uso de la vista (Activity) para rellenar los campos de un formulario, una vez que ha terminado de completar los campos le da al botón "Crear reto", que genera un evento(crear reto). Este evento se acompaña con los datos del reto encapsulados en un objeto *DTO*(TransferReto), que se pasa desde la vista (Activity) hacia el *controlador.* El *controlador* pasa el evento a la *factoría de comandos*, que se encarga de crear el *comando(crear reto)*. Dicho *comando* llama a la *factoría de servicios de aplicación* con el fin de emplear un *servicio de aplicación,* en este caso será el servicio de aplicación de sucesos que se encargará de recibir el *DTO*.

Este *servicio de aplicación* se comunicará con la capa de integración, para ello hará uso del DBHelper que debe estar asociado a un contexto, por lo tanto utilizamos el *singleton* para acceder al mismo y poder manejar el DAO(Reto). El *DAO* representa la tabla reto en nuestra base de datos. Para crear un reto el DAO se encarga de acceder a los campos del DTO mediante getters y el DBHelper de hacer la inserción en la tabla. Una vez que se aplican las reglas de negocio en el *servicio de aplicación*, éste se encarga de retornar al *comando*(crear reto) otro objeto *DTO* con la información necesaria para actualizar la vista.

A continuación, el *comando* pasa el *DTO* al *controlador* que es el responsable de pasar el evento(crear reto) y el DTO al *dispatcher*. Finalmente, el *dispatcher* actualiza la vista correspondiente haciendo uso del *singleton* para acceder al contexto y del DTO para mostrar la información necesaria, en este caso sería un mensaje de éxito: "El reto Lavarse las manos después de comer se ha añadido correctamente". La figura 32 y 33 muestran los diagramas de secuencia de la capa de presentación y negocio-integración para el evento crear reto.

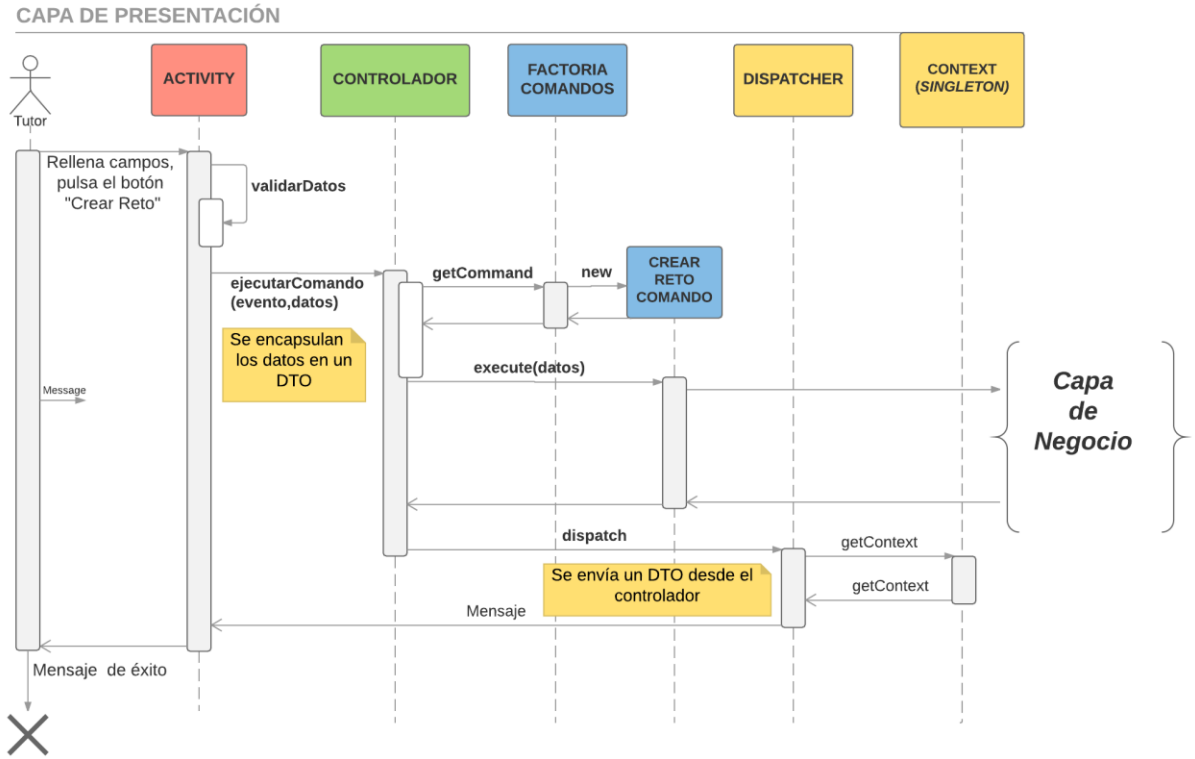


Figura 32: Diagrama de secuencia crear reto(presentación)

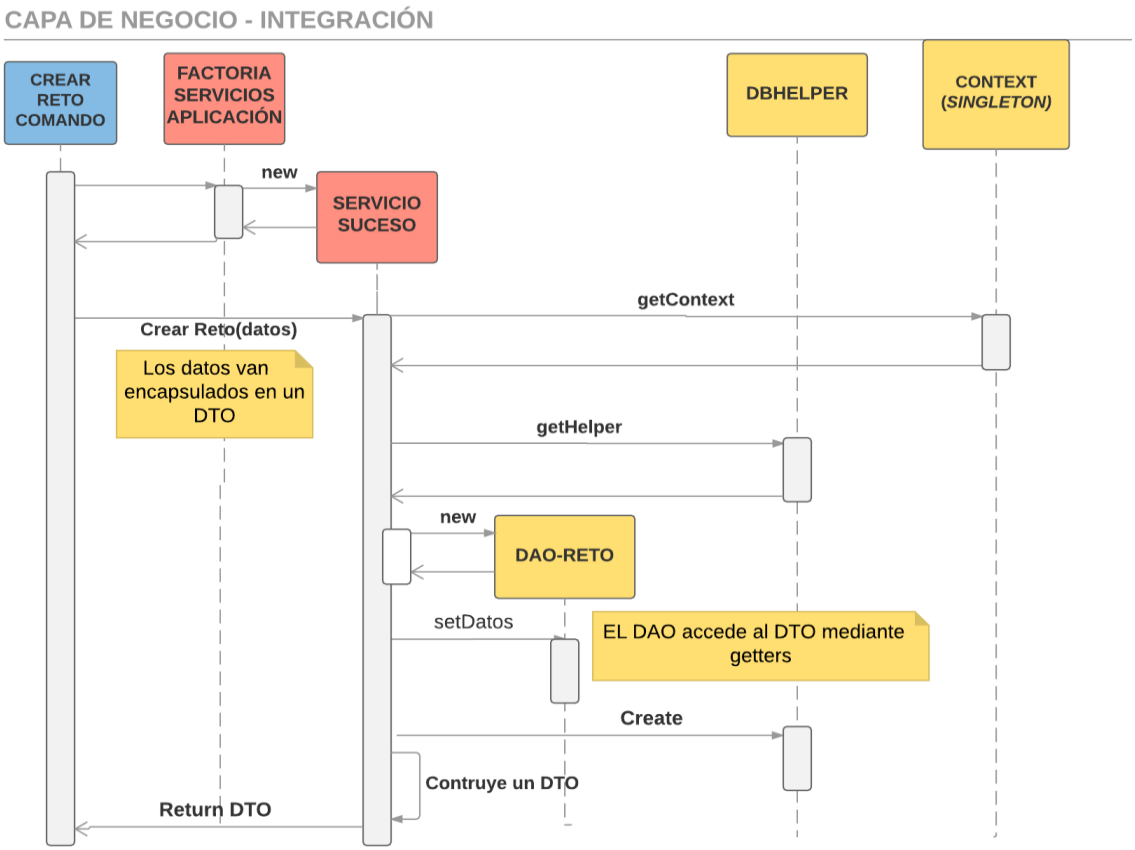


Figura 33: Diagrama de secuencia crear reto(negocio-integración)

5.2 Descripción de los patrones utilizados

El uso patrones de ingeniería de software nos proporcionan reusabilidad en el diseño y nos ofrecen ventajas a la hora de resolver problemas.

Para cada patrón que se ha utilizado se identifican 3 aspectos importantes:

- Las clases e instancias participantes.
- Los roles y colaboraciones de dichas clases.
- La distribución de responsabilidades.

➢ Singleton

Este patrón garantiza que sólo hay una instancia de una clase, proporcionando un único punto de acceso a ella. La fig. 34 muestra como es el diagrama de clases del patrón singleton.

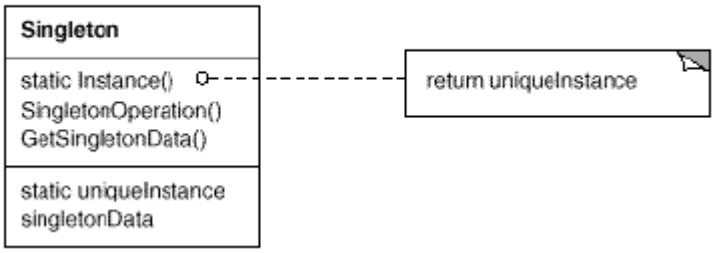


Figura 34: Diagrama de clases singleton

Hemos decidido usar este patrón porque teníamos dos problemas importantes: el acceso a la base de datos desde la capa de negocio y la actualización de la vista cuando ocurre un determinado evento, la causa de ambos problemas es que para realizar esas dos acciones se necesita el contexto, que es un atributo propio de las clases Activity. Para solucionarlos hemos decidido poner el contexto y otras características que también necesitábamos en un singleton de manera distinta en las aplicaciones tutor y usuario ya que su interfaz gráfica es diferente. La implementación del singleton en la *aplicación usuario* y tutor se puede ver en la figura 35 y 36 respectivamente.

```
public class Contexto {
    private static Contexto instancia = new Contexto();

    private Context actividadPrincipal;

    public static Contexto getInstancia() { return instancia; }

    public Contexto() {
    }

    public Context getContext() { return actividadPrincipal; }

    public void setContext(Context actividadPrincipal) {
        this.actividadPrincipal = actividadPrincipal;
    }
}
```

Figura 35: Singleton *aplicación usuario*

```
public class Manager {
    private static Manager ourInstance = new Manager();
    private AppCompatActivity activity;

    // Metodo que se accede a la instancia
    public static Manager getInstance() { return ourInstance; }

    // Constructor privado del singleton
    private Manager() { }

    // Metodo que cambia la activity, se utiliza desde las propias activities
    public void setActivity(AppCompatActivity miActivity) { activity = miActivity; }

    public Activity getActivity() { return activity; }
    // Metodo que devuelve el context, se utiliza desde los servicios de aplicacion
    public Context getContext() { return activity.getApplicationContext(); }

    // Metodo que devuelve el FragmentManager, se utiliza desde el dispatcher
    public android.support.v4.app.FragmentManager getFragmentManager(){
        return activity.getSupportFragmentManager();
    }
}
```

Figura 36: Singleton *aplicación tutor*

➢ Controlador de Aplicación

Patrón encargado de centralizar y modularizar la gestión de acciones y de vistas. En la figura 37 se muestra el diagrama de clases de este patrón.

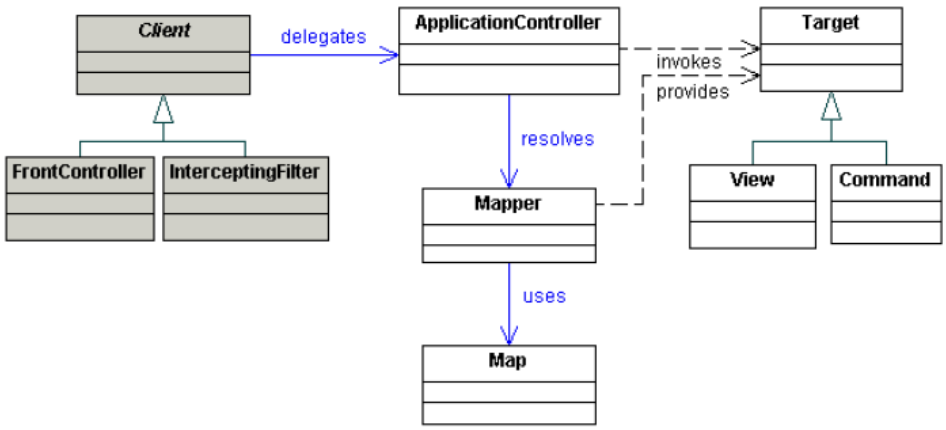


Figura 37: Diagrama de clases Controlador de Aplicación

Este patrón en las dos aplicaciones tutor y usuario se encarga de recibir los eventos que son lanzados principalmente por las activities y fragments. El controlador de aplicación hará uso del patrón command y una clase que tiene constantes para definir y estructurar cada evento. También tendrá encapsulado el dispatcher encargado de

actualizar la vista. La implementación del controlador para las aplicaciones tutor y usuario se puede ver en la figura 38.

```
public class ControladorImp extends Controlador {
    @Override
    public void ejecutaComando(String accion, Object datos) {
        Command comando = FactoriaComandos.getInstancia().getCommand(accion);
        Object ret;
        try {
            ret = comando.ejecutaComando(datos);
            actualizaVista(accion,ret);
        } catch (commandException e) {
            lanzarError(e.getMessage());
        }
    }
```

Figura 38: Implementación del controlador en el sistema AS

➢ Dispatcher

El patrón dispatcher es el encargado del control de la vista y la navegación, controlando la elección de la siguiente vista a mostrar. En la figura 39 se puede observar el diagrama de clases del dispatcher.

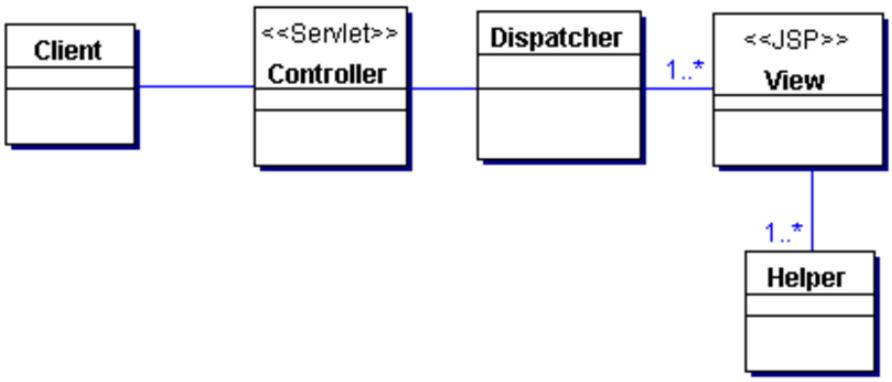


Figura 39: Diagrama de clases Dispatcher

Hemos usado este patrón en nuestro sistema porque queríamos resolver el problema de la centralización y manejo de la vista, queríamos tener un responsable que se encargara de actualizar el contenido de las activities o fragments. También usamos este patrón para conseguir mayor reutilización, modularidad y flexibilidad ya que mezclar la lógica del negocio con la de presentación nos podría traer confusiones en el código. El dispatcher lo hemos encapsulado dentro del controlador de la aplicación. Además, usa el singleton para lanzar las distintas actividades o fragments.

➢ Command

Este patrón permite solicitar una operación a un objeto sin conocer realmente el contenido de esta operación, ni el receptor real de la misma. Para ello se encapsula la petición como un objeto, con lo que además facilita la parametrización de los métodos. El diagrama de clases del patrón command se muestra en la figura 40.

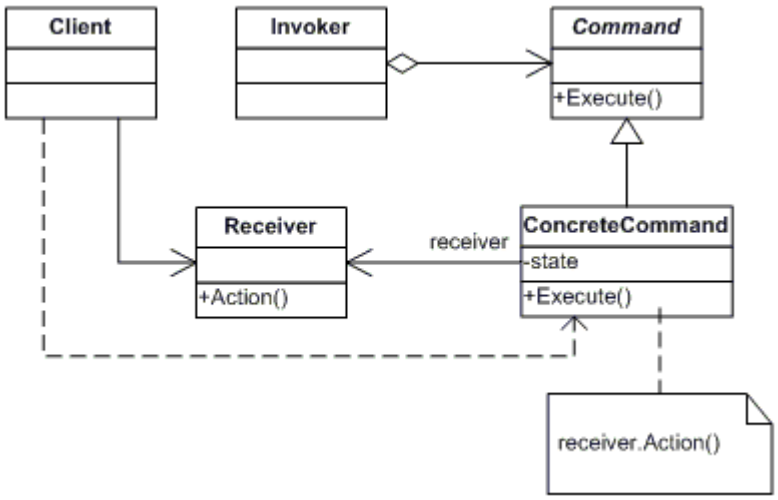


Figura 40: Diagrama de clases Command

Utilizamos este patrón porque nuestra aplicación se basa en órdenes. Cada uno de nuestros comandos está asociado a un evento que ocurre en la vista y que pasa por el controlador. Este patrón nos proporciona legibilidad, modularidad, sencillez y facilidad a la hora de depurar.

➢ Factoría Abstracta

El patrón *factoría abstracta* proporciona una interfaz para crear familias de objetos relacionados o que dependen entre sí, sin especificar sus clases concretas. Desliga la creación de objetos de la clase que se refiere a dichos objetos a través de la interfaz que implementan. En la figura 41 se puede observar el diagrama de clases de este patrón.

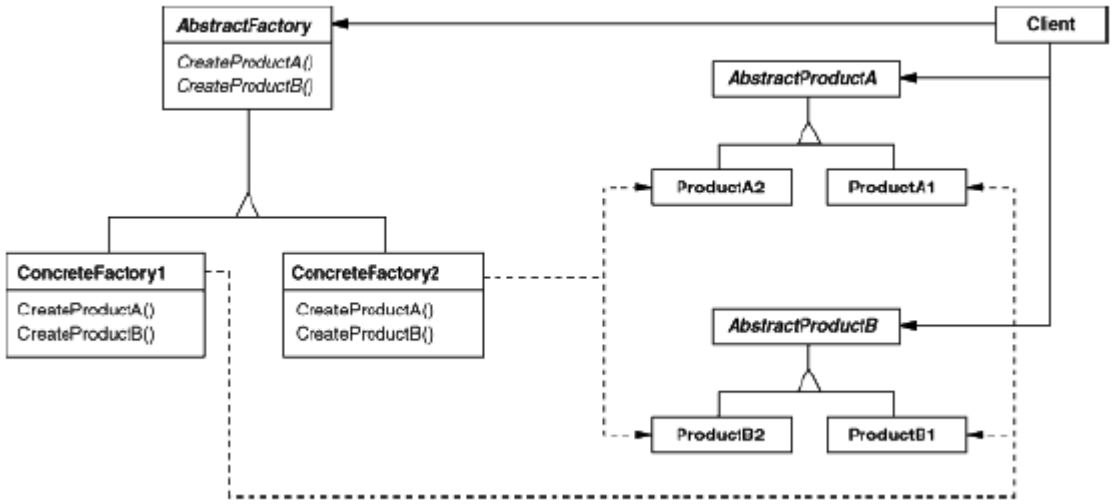


Figura 41: Diagrama de clases factoría abstracta

Este patrón lo hemos usado en la *aplicación usuario* y en tutor para crear familias de *Comandos* y familias de *Servicios de Aplicación*. Esto nos ha permitido aislar clases concretas consiguiendo una separación que nos facilita la depuración y modularidad de las clases involucradas y conseguir una mayor consistencia entre *Comandos* y *Servicios de Aplicación.*

➢ Data Transfer Object (DTO)

Este patrón permite independizar el intercambio de datos entre las distintas capas. Son objetos que no poseen una lógica de negocio, su único comportamiento es almacenar y entregar sus propios datos(accessors and mutators). El diagrama de clases del patrón DTO se puede observar en la figura 42.

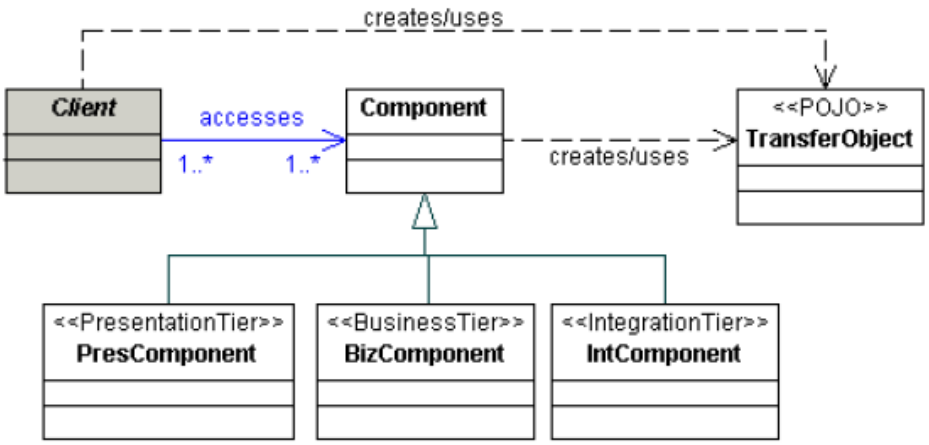


Figura 42: Diagrama de clases Data Transfer Object

En nuestro sistema los DTO son el mecanismo de comunicación entre las distintas capas de la arquitectura multicapa por tanto deben ser objetos serializados. Estos objetos no son idénticos en la *aplicación tutor* y la *aplicación usuario* puesto que tiene un comportamiento distinto y sus atributos no son los mismos en cada aplicación, sin embargo comparten algunos campos involucrados en la sincronización e intercambio de datos, para más información ver el capítulo 8 de este documento.

➢ Servicio de aplicación

El propósito principal de este patrón es centralizar la lógica de negocio, agrupar funcionalidades relacionadas, mejorar la reusabilidad del código y evitar la duplicación del mismo. En la figura 43 se muestra el diagrama de clases de este patrón.

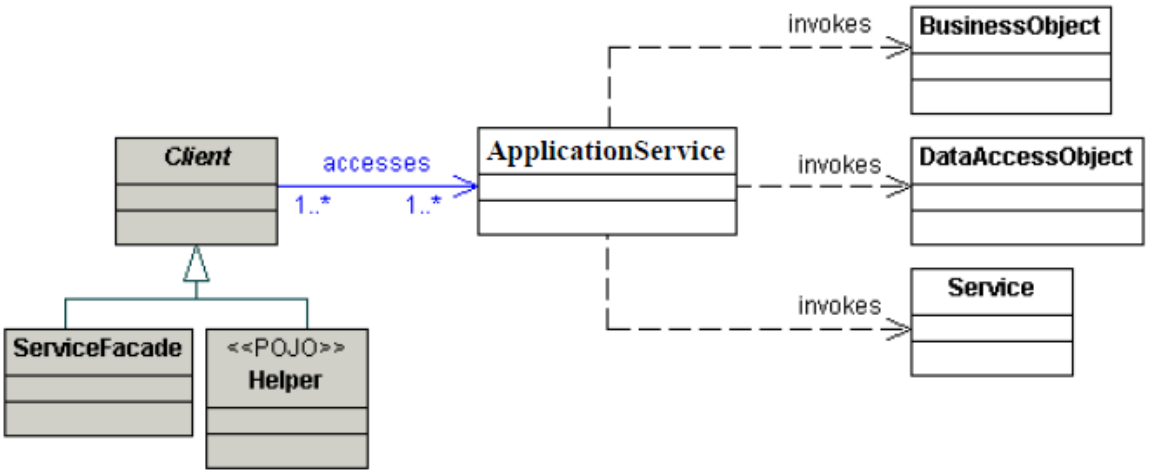


Figura 43: Diagrama de clases Servicio de Aplicación

Este patrón lo usamos en las dos aplicaciones del sistema AS pero de manera distinta ya que cuentan con una lógica de negocio diferente. Los servicios de aplicación en la *aplicación tutor* realizan operaciones CRUD para los usuarios, eventos, retos y tareas haciendo uso de los DAO y DTO, mientras que en la aplicación de usuario se utilizan principalmente operaciones de consulta.

➢ Data Access Object (DAO)

Permite acceder a la capa de datos, proporcionando representaciones orientadas a objetos a sus clientes que acceden a una base de datos o un archivo. El diagrama de clases del patrón DAO se observa en la figura 44.

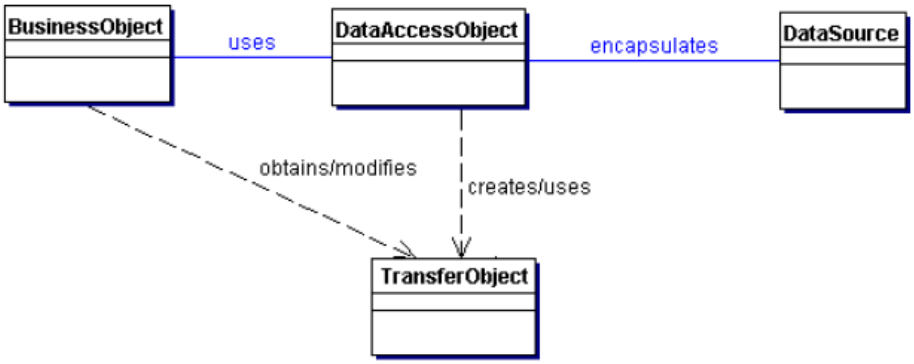


Figura 44**:** Diagrama de clases Data Access Object

En el sistema AS este patrón es de gran importancia ya que cada uno de nuestros *Data Access Object(DAO)* está mapeado con la librería ORMLite facilitando la manipulación de la base de datos tanto en la *aplicación usuario* como en la *aplicación tutor*. Para cada DAO se realiza un mapeo distinto dependiendo del tipo de datos que contenga la clase. Esto se consigue poniendo una serie de etiquetas sobre la clase que queremos que se represente como una tabla en la base de datos y sobre cada uno de sus atributos como se puede ver en la figura 45.

El funcionamiento de ORMLite consiste en crear una base de datos relacional y gestionar todas sus dependencias y relaciones una vez que se inicia la aplicación. Para el acceso y utilización de la base de datos hemos creado una clase DBHelper que proporciona métodos para la gestión todos los DAO, este funcionamiento se puede ver en la figura 46.

```
public class Usuario {

    @DatabaseField(generatedId = true, columnName = "ID")
    private Integer id;

    @DatabaseField(columnName = "NOMBRE")
    private String nombre;

    @DatabaseField(columnName = "CORREO")
    private String correo;

    @DatabaseField(columnName = "AVATAR")
    private String avatar;
```

Figura 45: Uso de la librería OrmLite

```
public TransferUsuario editarUsuario(TransferUsuario datos) {
    Dao<Usuario, Integer> daoUsuario;
    TransferUsuario ret = new TransferUsuario();
    try {
        daoUsuario = getHelper().getUsuarioDao();
        if (daoUsuario.idExists(1)) {
            Usuario usuario = daoUsuario.queryForId(1);
            usuario.setNombre(datos.getNombre());
            usuario.setAvatar(datos.getAvatar());
            usuario.setColor(datos.getColor());
            usuario.setTono(datos.getTono());
            daoUsuario.update(usuario);
        }else
            return null;
    }catch (SQLException e) {
        e.printStackTrace();
    }
    return datos;
}
```

Figura 46: Uso del DAO en el servicio de aplicación

CAPÍTULO 6: Aplicación usuario

"La simplicidad llevada al extremo
se convierte en elegancia."
JON FRANKLIN

En este capítulo vamos a explicar toda la funcionalidad y la arquitectura de la *aplicación usuario*, destinada a personas Asperger. En primer lugar, haremos una pequeña reflexión acerca de porqué elegimos el sistema operativo Android para desarrollar ambas aplicaciones, después definiremos una nomenclatura propia para todo el sistema AS, luego se detallará la estructura interna de la *aplicación usuario* así como su interfaz y las posibles acciones que puede llevar a cabo. Por último, explicaremos las tablas que forman su base de datos. La figura 45 muestra el esquema de la arquitectura global del sistema, resaltando en color la parte en la que se centra este capítulo

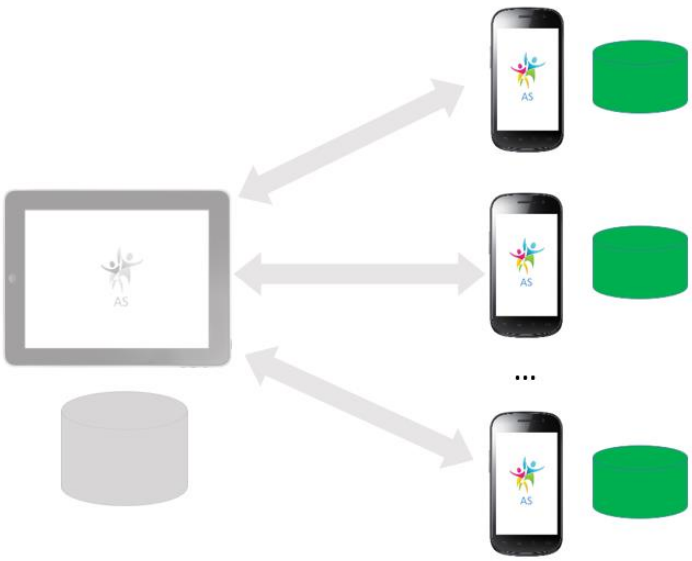


Figura 45: La *aplicación usuario* en el esquema de la arquitectura global del sistema

6.1 Elección sistema operativo Android

Cuando empezamos a plantear este proyecto queríamos, por una parte, que nuestras aplicaciones pudieran ser potencialmente utilizadas, ampliadas o mejoradas por el mayor

número de personas posibles; por otra parte, teníamos claro que queríamos hacer un desarrollo nativo de una tecnología móvil, dado que no habíamos tenido oportunidad de trabajar con ellas a lo largo de nuestro paso por la universidad y por tanto, nuestra elección para conseguir esos objetivos fue Android.

Según un informe de Kantar [4], Android era el sistema operativo con mayor cuota del mercado español en el 2015 (fig. 46).

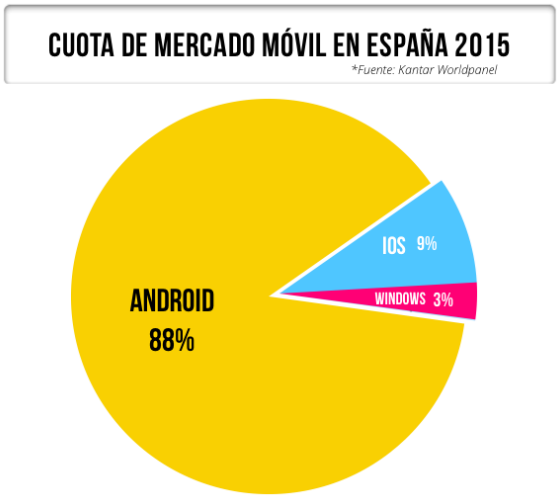
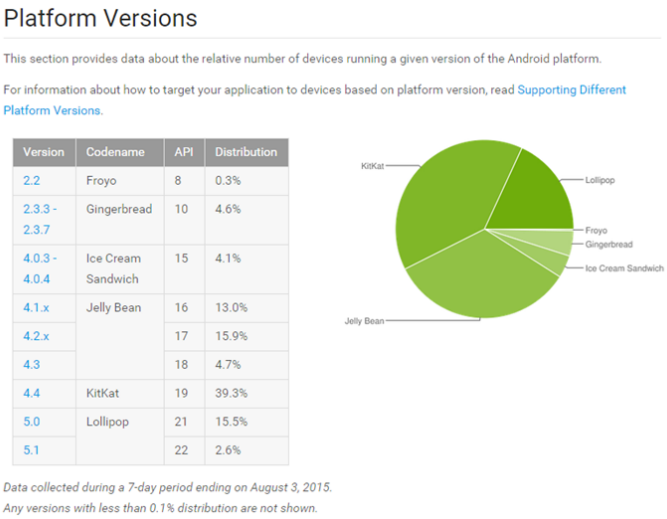


Figura 46: Cuota mercado móvil en España durante 2015

Cada cierto tiempo, Google realiza estadísticas acerca de las distintas versiones Android en funcionamiento [5, 6]. Teniendo en cuenta la tendencia de que las últimas versiones lanzadas son las que acaban siendo predominantes, inicialmente pensamos hacer las dos aplicaciones en la versión Lollipop, para que tuvieran más "tiempo de vida útil"; sin embargo, después del primer encuentro con nuestro cliente, pudimos observar que las versiones de los dispositivos que poseían los alumnos eran bastante inferiores y consecuentemente decidimos elegir la versión Kitkat, que era la predominante en ese momento (fig. 47), para implementar la *aplicación usuario*.



Platform Versions

This section provides data about the relative number of devices running a given version of the Android platform.

For information about how to target your application to devices based on platform version, read Supporting Different Platform Versions.

Version	Codename	API	Distribution
2.2	Froyo	8	0.3%
2.3.3 - 2.3.7	Gingerbread	10	4.6%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	4.1%
4.1.x	Jelly Bean	16	13.0%
4.2.x		17	15.9%
4.3		18	4.7%
4.4	KitKat	19	39.3%
5.0	Lollipop	21	15.5%
5.1		22	2.6%

Data collected during a 7-day period ending on August 3, 2015.
Any versions with less than 0.1% distribution are not shown.

Figura 47: Estadísticas uso versiones Android (agosto 2015).

6.2 Nomenclatura

En esta sección definimos una serie de conceptos propios de ambas aplicaciones para facilitar la comprensión de las secciones y capítulos posteriores.

- *Usuario*: persona que interactúa con la *aplicación usuario*, alumno Asperger. Un usuario está definido por los siguientes campos:
  - Datos personales: nombre, correo, teléfono, DNI, dirección y notas. Estas últimas contienen información extra acerca de algún asunto relacionado con el usuario como alergias, medicamentos, etc.
  - Datos de sus progenitores: nombre del padre, nombre de la madre, teléfono del padre, teléfono de la madre, correo del padre y correo de la madre.
  - Datos académicos: curso y centro académico.
  - Datos personalizables de la *aplicación usuario*: avatar, color y tono con el que el usuario quiere recibir sus notificaciones.
  - Puntuación.
  - Puntuación anterior, a la última tarea contestada.
  - Código de sincronización, clave alfanumérica necesaria para el intercambio de información entre aplicaciones.
  - Tipo de perfil, ver la descripción de “Perfil” al final de este subapartado.
- *Tutor*: persona que interactúa con la *aplicación tutor*, profesor especialista. Un tutor está definido por los siguientes campos:
  - Nombre.
  - Correo.

- ➢ Contraseña, es la clave de acceso a la *aplicación tutor*.
- ➢ Código de sincronización, es un conjunto de caracteres (las tres primeras letras del nombre del tutor) que se emplean para crear los códigos de sincronización de los usuarios y de esta manera asociarlos al tutor.
- ➢ Pregunta, es una cuestión de seguridad en caso de olvido de la contraseña.
- ➢ Respuesta, es la solución a la pregunta de seguridad.

❖ *Tarea*: actividad rutinaria que una persona Asperger suele olvidar y por tanto la *aplicación usuario* se la recuerda, con el fin último de que la vaya interiorizando y pueda llegar a pueda realizarla por sí solo. El profesor especialista debe conocer a la perfección la rutina de su alumno para poder crear baterías de preguntas con recordatorios motivantes y con las horas más adecuadas. Una tarea está definida por los siguientes campos:

- ➢ Texto de la alarma.
- ➢ Hora de la alarma, campo que almacena la hora y fecha de cuando debe aparecer la alarma.
- ➢ Texto de la pregunta.
- ➢ Hora de la pregunta, campo que almacena la hora y fecha de cuando debe aparecer la pregunta.
- ➢ Frecuencia de la tarea, indica la periodicidad con la que la tarea es recordada (diaria, semanal o mensual).
- ➢ Mejorar, es el umbral a partir del cual la frecuencia desciende.
- ➢ Datos para el seguimiento
  - ■ Contador.
  - ■ Número de “Si”, es una variable en donde se acumulan el número de respuestas positivas para la pregunta.
  - ■ Número de “No”, es una variable en donde se acumulan el número de respuestas negativas para la pregunta.
  - ■ No seguidos, es una variable en donde se almacenan el número de respuestas negativas consecutivas, si esta llega a tres, la frecuencia aumenta.
  - ■ Habilitada, variable que indica si la pregunta está activa o no.
- ➢ Notificación de la alarma, guarda el identificador de la alarma en caso de que se necesite editar o eliminar.
- ➢ Notificación de la pregunta, guarda el identificador de la pregunta en caso de que se necesite editar o eliminar.

- ➢ *Reto*: cometido que se presenta como un desafío con una recompensa opcional para que el usuario elimine un defecto o corrija un hábito. Un reto tiene los siguientes campos: texto (una descripción del reto), premio, contador del progreso y una variable que indica si se ha superado el reto.
- ➢ *Evento*: acontecimiento puntual, al cual el tutor invita a determinados usuarios. Los campos de un evento son: nombre, fecha, hora de la alarma, hora del evento y asistencia.

- ❖ *Suceso*: término general para reto, evento o tarea.
- ❖ *DAO (Data Access Object)*: componente software para manejo de datos entre la aplicación y la base de datos. (Más información en el capítulo 5 y el anexo A de este documento).
- ❖ *DTO (Data Transfer Object):* componente software para el intercambio de información entre las diferentes capas de la aplicación. (Más información en el capítulo 5 y el anexo A de este documento).
- ❖ *Perfil*: es un modo de agrupar una serie de tareas para determinados grupos de personas. En esta versión existen tres tipos de perfil: el perfil "estándar", que contiene un mínimo de tareas comunes a todos los usuarios; el perfil "chica", con todas las tareas del perfil "estándar" además de algunas específicas para el género femenino; y el perfil "vacío" que no contiene ninguna tarea predefinida.

6.3 Estructura interna de la aplicación

Como ya se ha explicado en el capítulo 5 de este documento, con el objetivo de separar las reglas de negocio de la interacción directa con el usuario decidimos desarrollar la aplicación con una arquitectura multicapa e implementarla usando patrones de software. Como consecuencia de esta decisión la organización en paquetes de las clases Java quedó configurada como se muestra en la fig. 48 .

La clase que está dentro del paquete de integración es la que permite la conexión a la base de datos, junto con manejo de datos de la misma. El paquete de negocio contiene las clases necesarias para la correcta comunicación entre las vistas y la aplicación, desde una perspectiva basada en el desarrollo software. En este paquete también se encuentran las clases que establecen la conexión entre los dos tipos de aplicación. Por último el paquete de presentación está subdivido en tres: el controlador, que engloba los comandos y ciertas clases moderadoras; las notificaciones, que poseen las clases que se ocupan tanto de la ejecución como de la recogida

de las respuestas por parte de las notificaciones; y la vista, que contiene todas las clases que extienden la clase Activity de Android.

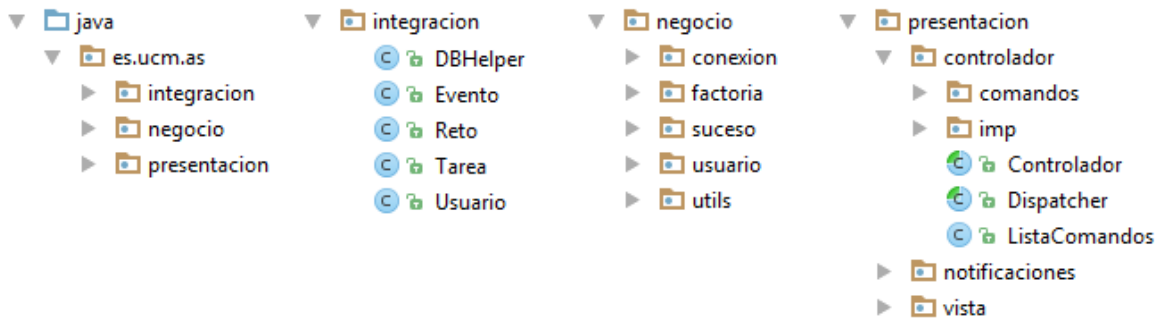


Figura 48: Organización en paquetes de las clases Java en la *aplicación usuario*.

Para más información del diseño interno de la aplicación, se recomienda leer los diagramas de clases que se ilustran detalladamente en el anexo A de este documento.

6.4 Interfaz y funcionalidad de la aplicación

Dado que una de nuestras metas era conseguir una aplicación que no estuviera enfocada a un público infantil hemos procurado implementar una interfaz sencilla y sobria con la que cualquier persona pueda interactuar cómodamente. Como se puede apreciar en la fig. 49, todas las pantallas se dividen en dos partes: cabecera (parte verde) y cuerpo (parte azul).

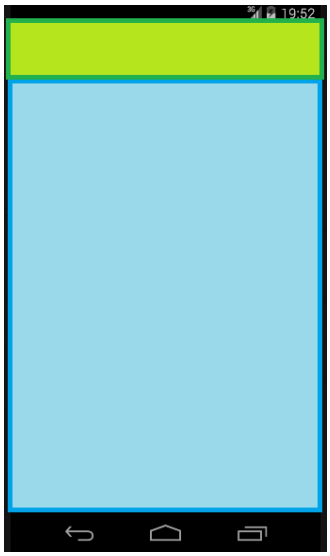


Figura 49: Esquema de las pantallas en la *aplicación usuario*

En la cabecera, se encuentran a la izquierda los iconos para la navegación entre pantallas y a la derecha el icono de ayuda. Como se puede apreciar en la fig. 50.

Figura 50: Ejemplos de cabeceras en la *aplicación usuario*

Esta aplicación tiene como objetivo ayudar a que las personas Asperger, que hagan un correcto uso de ella, recuerden sus actividades cotidianas con la intención de que adquieran hábitos por sí mismas. Su funcionalidad está destinada a que un usuario realice determinadas tareas, que se le avisan mediante una notificación tal y como aparece en la parte izquierda de la fig. 51, y después de un breve periodo de tiempo debe contestar una pregunta indicando si ha realizado o no dicha tarea, esta pregunta también es realizada mediante una notificación tal y como se muestra en la parte derecha de la fig. 51.

Figura 51: Ejemplo de las notificaciones que recuerdan una acción y preguntan si se ha realizado, respectivamente.

La asiduidad con la que aparece una tarea viene en función de su frecuencia y de su umbral de mejora. Por defecto la frecuencia es diaria y el umbral es contestar "sí" treinta veces la pregunta de la tarea. En función de las respuestas del usuario la frecuencia se ve alterada: en caso de que conteste "no" tres veces seguidas, la frecuencia de la tarea aumenta; por el contrario, si contesta "sí" un número de veces igual al umbral, la frecuencia de la tarea desciende. Aumentar implica el paso de mensual a semanal y de semanal a diaria, porque si la tarea ya tiene una frecuencia diaria no puede incrementarse más; descender conlleva lo contrario. Es decir que, si el usuario realiza de forma continuada la tarea, progresivamente se le recuerda menos y en caso de que deje de realizarla durante tres días seguidos hay que volver a preguntarle con mayor continuidad. Este procedimiento fue acordado con la Asociación Implica para aseguramos de la correcta interiorización de la tarea por parte del usuario.

A la hora de cambiar de frecuencia el número de repeticiones necesarias para mejorar se ha establecido de manera fija. Si es semanal se repite 6 semanas, si es mensual se repite 3 meses y

si es diaria 30 días. Además, a petición del cliente los fines de semana no se ejecutará el sistema de notificaciones debido a las características académicas de la mayoría de las tareas.

A la hora de implementar las notificaciones tuvimos varios problemas para que se ejecutarán correctamente. En Android se pueden lanzar notificaciones en un momento exacto del día presente o se pueden mostrar en un instante cercano al establecido durante una frecuencia fijada, esto último no nos favorecía ya que necesitábamos que tanto las alarmas como las preguntas aparecieran cuando debían y por tanto al elegir la otra opción debimos desarrollar una manera para que las notificaciones pudieran mostrarse no solo un día sino varios de forma automática. Mediante un servicio, durante la madrugada, se leen todas las tareas correspondientes al día actual, luego en la última hora del día se cambian, en función de su frecuencia, la fecha de todas las tareas que han aparecido hoy para que en la próxima carga de tareas se localicen correctamente. De este modo se crea un bucle para lanzar las notificaciones cuando deben.

Cuando empezamos a desarrollar la *aplicación usuario* tuvimos que descartar nuestra primera idea sobre cómo instalarla en los dispositivos, ya que creíamos que era posible que la *aplicación tutor* generase la APK de la *aplicación usuario*, se la enviase por correo al usuario y este se la instalará en su dispositivo. Debido a que por lo general un móvil no dispone de un compilador que pudiese generar un APK, a que no se puede acceder al GCC interno del móvil y a los problemas de autenticación de la APK por no provenir de un sitio oficial se decidió cambiar la manera de generar e instalar la *aplicación usuario*. La solución final por la que se optó es que la *aplicación usuario* se descarga directamente de Google Play.

Cuando el usuario se instala la aplicación en el dispositivo, esta no dispone de ningún dato y solo se permite al acceso a ella si se ha registrado correctamente, para lo cual necesita un código de sincronización. Este código es generado por la *aplicación tutor*, cuando el tutor añade un usuario, y gracias a él se consigue una comunicación entre ambas aplicaciones para el intercambio de datos de las entidades suceso. De esta manera, se evita que alguien pueda descargarse la *aplicación usuario* y utilizarla sin estar asociado a una *aplicación tutor*. El aspecto de la sincronización se detalla en el capítulo 8.

Una vez registrado aparece la pantalla principal tal como se muestra en la fig. 52. Desde esta pantalla el usuario puede personalizar aspectos de la aplicación, consultar la pantalla de ayuda, ver su puntuación, sincronizar con la *aplicación tutor*, consultar sus tareas, consultar los eventos a los que está invitado, realizar su reto y enviar informes de su actividad.

Figura 52: Vista de la pantalla principal en la *aplicación usuario*.

A continuación, se describen con más detalle cada una de las funcionalidades:

- ***Configuración***

  Para personalizar algunos aspectos de la aplicación se debe clicar en el engranaje, situado en la parte derecha de la cabecera, en ese momento la pantalla cambia a la vista de configuración (fig. 53) desde la cual el usuario puede modificar: su nombre de usuario, el color de fondo de la aplicación, el tono con el que recibe las notificaciones y añadir una imagen para que la interacción sea más amigable. Estos cambios no son visibles para el tutor ni interfieren con el objetivo principal de nuestro proyecto.

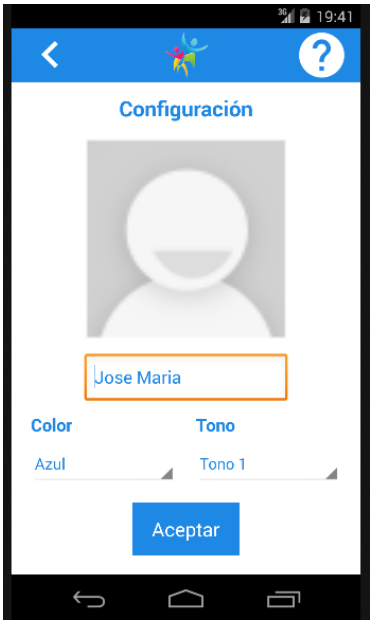


Figura 53: Vista de la pantalla de configuración en la *aplicación usuario*.

➢ ***Ayuda***

Cuando el usuario pulsa sobre el icono que está a la izquierda de la cabecera aparece una imagen de la pantalla principal en donde se explican cada uno de los elementos que la componen. En un principio se quería mostrar la ayuda en formato PDF pero debido a las incompatibilidades de Android con este formato de fichero era necesario abrir otra aplicación para visualizarlo, bien fuera una aplicación específica previamente instalada en el móvil o el navegador. Para evitar esto, ya que los dispositivos de los usuarios pueden ser lentos y el cambiar de aplicación es un proceso pesado, se decidió mostrar la ayuda para cada pantalla específica en formato imagen, un ejemplo sería la fig. 54; dicha imagen se puede ampliar mediante multitouch.

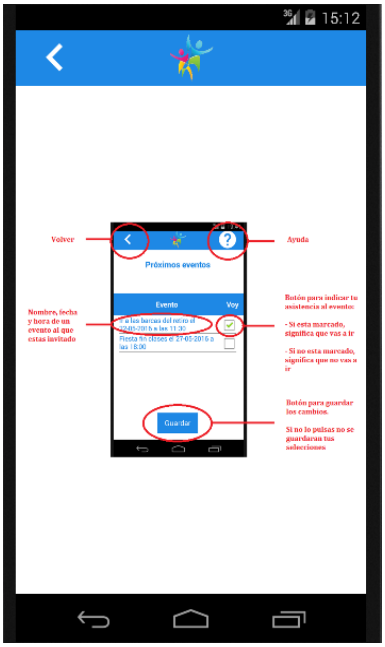


Figura 54: Vista de una pantalla de ayuda en la *aplicación usuario*.

➢ ***Puntuación***

La puntuación es un valor numérico que refleja la evolución del usuario en función del número de respuestas que ha contestado positivamente; su cálculo se lleva a cabo mediante la siguiente fórmula:

$$\text{Puntuación} = (10 / N) * \left(\sum_{i=1}^{N} pi\right) \qquad 0 \le i < N$$

$$p_i = 1 \quad \text{si } b_i >= 0$$

$$p_i = 0 \quad \text{si } b_i < 0$$

Siendo:

b_i = balance entre el número de respuestas positivas (+1) y negativas (-1) de la tarea i ésima.

N = número total de tareas.

La puntuación se recalcula al iniciar la aplicación, dado que el contenido de las respuestas se almacena en segundo plano, lo que indica que puede cambiar entre acceso y acceso a la aplicación.

➢ ***Sincronización***

Es un botón que permite la actualización de información entre el usuario y el tutor. Esta funcionalidad está descrita en el capítulo 8 de este documento.

➢ ***¿Cómo vas?***

Como hemos mencionado antes, las tareas buscan interiorizar un comportamiento en el usuario, para lograr este fin, primero se lanza una notificación recordando al usuario la acción que debía realizar a una determinada hora y unos instantes después se lanza otra notificación preguntándole si ha finalizado con éxito o no dicha acción. Este ciclo se repite en base a la frecuencia que tiene dicha tarea (diaria, semanal o mensual).

Cuando se selecciona esta opción aparece una pantalla, como la de la fig. 55 en la cual se muestra su puntuación actual, su puntuación previa a la última tarea contestada, una flecha que indica si ha habido una mejora o un empeoramiento entre ambas y debajo hay una tabla donde se muestran todas sus tareas junto con el número de respuestas afirmativas o negativas que realizado de ellas y el balance total entre ambas.

Figura 55: Vista de la pantalla "¿Cómo vas?" de la *aplicación usuario*.

➢ ***Próximos eventos***

Esta funcionalidad está destinada a facilitar la organización y recoger la asistencia de los usuarios a ciertos acontecimientos como fiestas o salidas programadas. Los usuarios

al pulsar la opción “Próximos eventos” pueden ver la lista de eventos a los que están invitados y confirmar su asistencia.

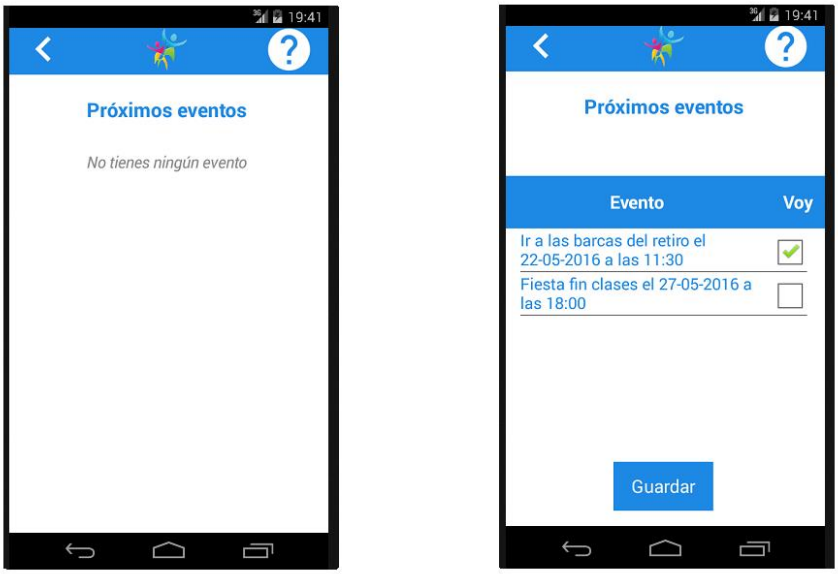


Figura 56: Vistas de las posibles pantallas “Próximos eventos” en la *aplicación usuario*.

➢ ***Reto***

Como se ha mencionado anteriormente los retos son desafíos para determinados usuarios y por tanto no todos tienen uno. En la fig. 57 se pueden observar las pantallas que le aparecerian a un usuario sin reto y a otro con uno asignado.

Las diferencias entre una tarea y un reto son varias: el reto no tiene frecuencia, lo que implica que su duración no está establecida; tampoco existe la obligación de contestarlo, es decir, que el hábito que se pretende interiorizar en el usuario no es prioritario; y a diferencia de las tareas, las respuestas del reto no se consiguen mediante una notificación, es el usuario quien debe entrar en la aplicación a contestar su reto.

Figura 57: Pantallas de la opción "Reto" en la *aplicación usuario*

Una vez finalizado el reto, en otras palabras, cuando el usuario ha contestado afirmativamente treinta veces, los botones desaparecen para que no pueda seguir contestando, el reto se marca como superado y se queda a la espera de la siguiente sincronización con el tutor.

➢ ***Enviar informe por correo***

Inicialmente se pretendía que la *aplicación usuario* enviará automáticamente un informe al correo del usuario con una frecuencia elegida por el mismo. Dado que no se dispone de un servidor, la automatización de correos requiere una autenticación desde el propio móvil, para la que se necesita la contraseña del correo electrónico que se usa para enviar. Se valoraron las siguientes posibilidades:

a. Enviarlo desde el propio correo del usuario. El inconveniente es que habría que pedirle la contraseña y, por tanto, almacenarla en la BBDD local. Dado que ésta no está cifrada ni dispone de medidas de seguridad suficientes para guardar información tan sensible, se descarta.
b. Creación de una cuenta de uso expreso para la aplicación, común para todos los usuarios y de la que nosotros, como desarrolladores, disponemos de la contraseña (as.noreply.pc@gmail.com). Esta segunda solución también supone serios problemas para la privacidad de los usuarios. A todos sus informes se podría acceder desde la carpeta "Enviados" de esta cuenta, además la

contraseña aparecería en el código y dado el carácter abierto de este proyecto, esta solución es inviable.

Se propuso como solución descartar el tema de la automatización y enviar el correo desde una de las aplicaciones de mensajería del dispositivo cuando el usuario pulsará la opción “Enviar informe por correo”. Al hacerlo se abre un ventana con las distintas aplicaciones de email disponibles en el dispositivo; cuando se selecciona una, aparece un texto y un asunto predefinidos por nosotros, el informe como un archivo adjunto y como destinatario se encuentra la dirección de correo proporcionada en el registro. Se requiere la interacción de la persona física ya que de esta manera se evita el acceso, el uso y el manejo de contraseñas personales.

6.5 Base de datos

Esta sección está destinada a explicar los detalles de la base de datos de la *aplicación usuario*. Como se ha mencionado anteriormente, esta aplicación solo permite la existencia de un único usuario, el cual puede modificar sus datos pero no le está permitido crear, eliminar o modificar ninguna entidad suceso. Es decir que toda la información almacenada estará relacionada exclusivamente con ese usuario, por esta razón decidimos que no existiera ningún vínculo entre las tablas, de esta manera conseguimos eliminar la redundancia de datos.

Dado que no tenemos servidor y la base de datos es local, si esta se elimina o se corrompe no hay posibilidad de recuperación. A continuación, se detallan las cuatro tablas que almacenan la información de los distintos DAO’s:

- *Tarea*: almacena las tareas que el tutor asigna al usuario con toda la información necesaria para poder generar sus recordatorios, sus preguntas, almacenar sus respuestas y realizar su seguimiento.
- *Reto:* contiene los datos relativos al reto asignado por el tutor.
- *Evento*: almacena aquellos eventos a los que el tutor ha invitado al usuario así como si va a asistir o no a ellos.
- *Usuario*: contiene la información relativa al perfil del usuario, cuya información más valiosa es la puntuación, puesto que el resto de campos solo son visibles para el propietario del dispositivo.

En la fig. 58 se puede ver el diagrama que ilustra el modelo entidad relación, en el que se destacan los atributos que son clave primaria.

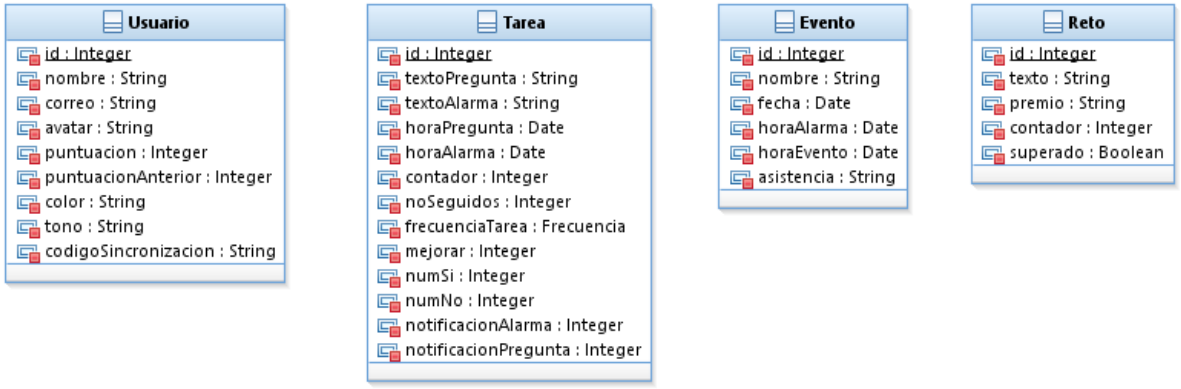


Figura 58: Diagrama entidad relación de la *aplicación usuario*

CAPÍTULO 7: Aplicación tutor

"La inteligencia consiste no sólo en el conocimiento,
sino también en la destreza de aplicar
los conocimientos en la práctica."
ARISTÓTELES

En este capítulo vamos a explicar en detalle la *aplicación tutor*, diseñada para un profesor especialista que trata con alumnos Asperger y quiere usar el sistema informático AS. El objetivo de esta aplicación es que el profesor pueda tener centralizada toda la información de sus alumnos, gestionar los sucesos de cada uno de ellos (tareas, eventos y retos) y disponer de un seguimiento preciso de su evolución.

En la fig. 59 se ha resaltado en el diagrama de la arquitectura global del sistema lo correspondiente a la *aplicación tutor*: una aplicación Android diseñada para tablet, aunque escalable para poder ser utilizada en otros dispositivos móviles más pequeños, y una base de datos local propia de la aplicación. Como se puede ver en la figura tan solo habrá una *aplicación tutor* en un sistema AS.

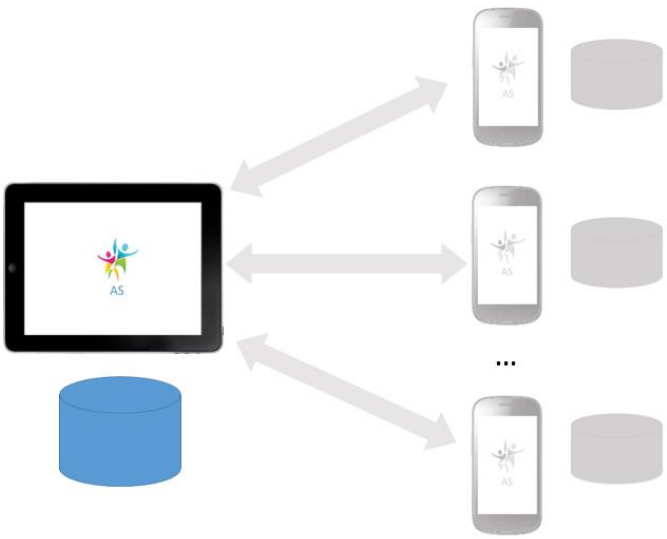


Figura 59: Esquema de la arquitectura global del sistema (*aplicación tutor*)

En este capítulo presentaremos la estructura interna, la interfaz y la funcionalidad de la aplicación. Después, explicaremos las tablas y la arquitectura de la base de datos. La nomenclatura usada en la *aplicación tutor* es la misma que en la *aplicación usuario* (ver sección 6.2)

7.1 Estructura interna de la aplicación

Al igual que en la *aplicación usuario*, para el desarrollo de la *aplicación tutor* hemos utilizado los patrones de ingeniería del software que se explicaron en detalle en el capítulo 5. Como se puede ver en la fig. 60, la organización en paquetes de las clases Java queda dividida en tres tipos de paquetes: integración, negocio y presentación. El paquete de integración tiene la clase dedicada a la base de datos y los DAOs *(Data Access Objects)*. El paquete de negocio contiene los DTOs *(Data Transfer Objects)*, los servicios de aplicación y las clases que intervienen en la conexión entre esta aplicación y la *aplicación usuario*. El paquete de presentación está a su vez subdividido en dos: por una parte las clases asociadas a las vista que son las que que extienden de la clase Activity de Android y por otra parte, las clases que forman el controlador, es decir, que implementan la funcionalidad como la clase controlador, el dispatcher y los comandos.

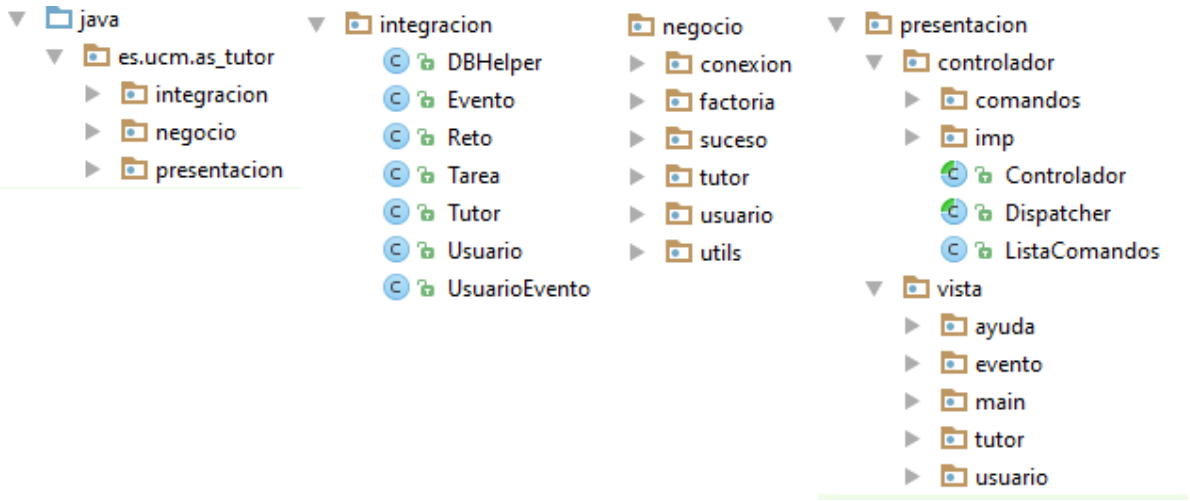


Figura 60: Organización en paquetes de las clases Java en la *aplicación tutor*

Para comprender bien el funcionamiento interno de esta aplicación se recomienda leer el anexo A de este documento donde se muestran los diagramas de clases de las entidades con los atributos y métodos que poseen.

7.2 Interfaz y funcionalidad de la aplicación

La interfaz se ha diseñado siguiendo el último estilo implantado por Google, Material Design [7], lo cual hace que sea más vistosa sin perder su sencillez. Se tomó esta decisión principalmente porque los componentes de desarrollo Android que se ofrecen a partir de la API 21 [8] nos permiten mostrar las múltiples opciones que ofrece esta aplicación de una manera muy limpia e intuitiva.

Los diferentes tipos de menú de navegación (Menu items, drawers, toolbars), los botones flotantes que prevalecen encima de fragments (Floating action buttons), los diálogos (Dialogs), los controles de selección (Checkbox, radio button, switch, spinner), los campos de texto (Floating inline labels) y los recogedores (Date pickers, time pickers) son algunos de los elementos que hemos usado en la aplicación y que ya vienen diseñados con efectos y estilos de manera que mejoran la experiencia de usuario.

Para conseguir consistencia entre las distintas secciones de la aplicación decidimos usar fragments. Un fragment [9] representa el comportamiento de una porción de la interfaz en una activity [10] , que es la ventana con la que el usuario interactúa. En la fig. 61 se ilustra cómo la activity principal de la aplicación se divide en dos fragments dinámicos independientes, a la izquierda (naranja) se mostrarán los listados y a la derecha (rojo) el detalle del elemento seleccionado. En la parte superior de la figura se ha destacado en rosa la barra de navegación que estará presente en todas las pantallas de la aplicación aunque con distintos elementos ya que el menú y los elementos que aparecen varían para cada sección.

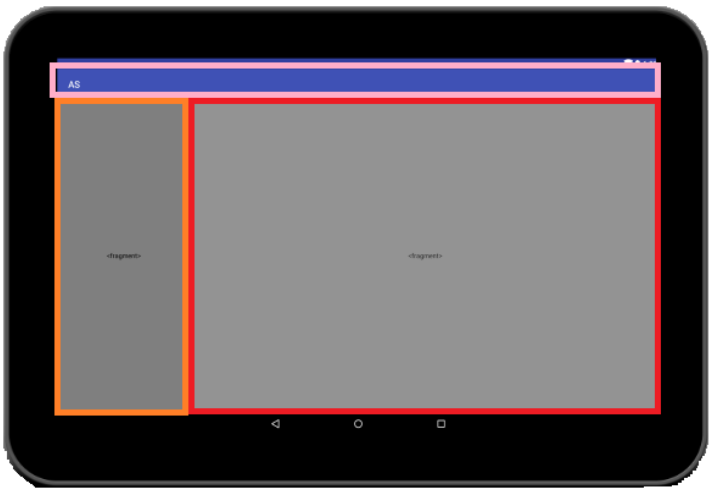


Figura 61: Esquema de la activity principal de la *aplicación tutor*

La aplicación cuenta con un menú en el costado lateral (Drawer) que recoge los cuatro módulos principales de la aplicación tal y como se muestra en la fig. 62: usuarios, eventos, mi perfil y ayuda. Este menú aparece con un efecto deslizante desde la izquierda de la pantalla y desaparece en cuanto el usuario selecciona una opción. Es siempre accesible desde la barra de navegación superior. A continuación, explicaremos cada una de las opciones del menú.

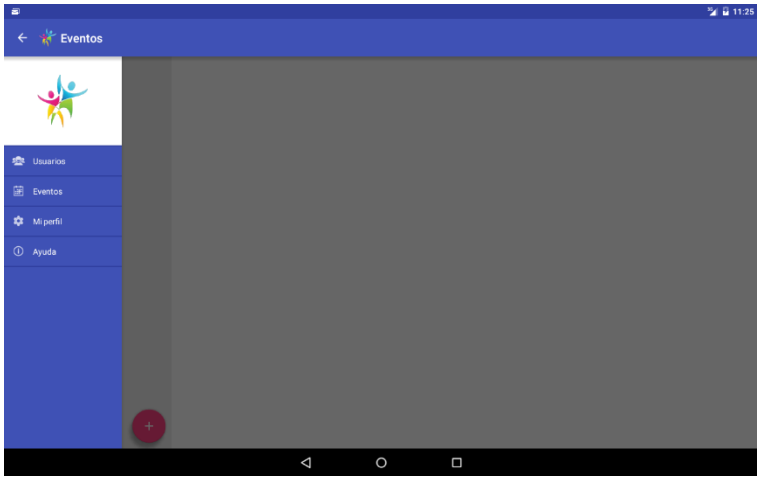


Figura 62: Menú lateral de la *aplicación tutor*

Antes de comenzar, en la fig. 63 se muestra la pantalla de registro en la que el profesor especialista debe introducir sus datos para el buen funcionamiento de la aplicación después de instalar la aplicación por primera vez. Se le solicita su nombre, su correo electrónico, una clave de acceso y una pregunta de seguridad para recuperarla con la respuesta. Todos los campos son obligatorios.

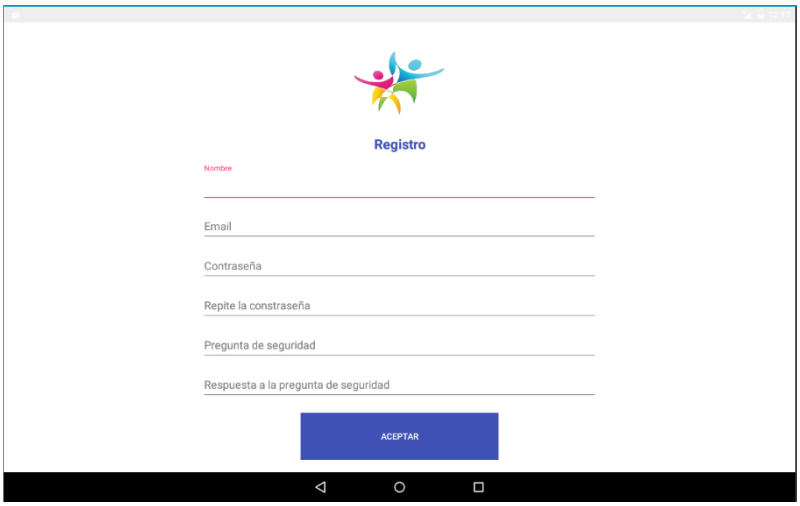


Figura 63: Pantalla de registro de la *aplicación tutor*

El primer módulo es **"Usuarios"**, al seleccionar esta opción aparece la pantalla que se muestra en la fig. 64. A la izquierda se puede ver el listado con todos los usuarios dados de alta en el sistema, en el centro de la pantalla se muestra la información del usuario seleccionado con todos sus campos editables por lo que se puede editar un usuario y guardar los cambios en cualquier momento.

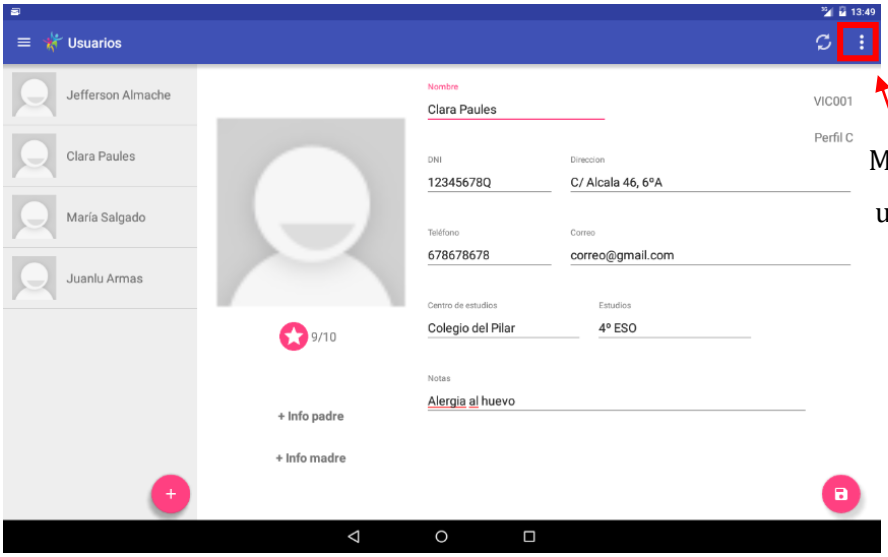


Figura 64: Vista de la sección de usuarios de la *aplicación tutor*

En la parte inferior izquierda hay un botón que permite crear un usuario nuevo. Para crear un nuevo usuario el profesor completa la información que desee ya que sólo dos campos son obligatorios: el nombre del usuario y el perfil. El objetivo de asignar un perfil a cada usuario es que se le añaden por defecto una serie de tareas que son comunes a la mayoría de usuarios, esto supone una facilidad para el profesor, que así dispone de una base sobre la que programar las tareas específicas del nuevo usuario en vez de comenzar de cero.

Una vez completados los campos del nuevo usuario se debe seleccionar el botón de guardar situado en la parte inferior derecha de la pantalla. La aplicación se encarga de cargar en la BBDD la batería de preguntas leyendo del fichero correspondiente al perfil. Esto se ha realizado a través de la implementación de una clase "ParserText" capaz de transformar cadenas de caracteres en los distintos campos que tiene una tarea como las horas de alarma y pregunta o la frecuencia.

Es importante señalar que cuando se añade un nuevo usuario al sistema, se genera su código de sincronización que es único y está formado por las tres primeras letras del nombre del tutor y unos dígitos (generados a partir del contador de usuarios en el sistema). Este código aparece en la parte superior derecha de la información del usuario junto con el perfil y no es editable. Para que un alumno pueda registrarse en la *aplicación usuario* se necesita este código por lo que el profesor debe comunicárselo.

En la parte superior derecha de la barra de navegación aparecen dos opciones relacionadas con el usuario seleccionado: un icono para realizar la sincronización y un menú propio del módulo usuarios.

Cuando el profesor selecciona el icono de sincronización  se inicia la comunicación con la *aplicación usuario* en caso de que ambos dispositivos se encuentren en la misma red WiFi tal y como se explica en el capítulo 8. Mediante este proceso se sincroniza la información de ambos dispositivos. Esto es importante porque realmente **aunque el tutor haga cualquier modificación en la *aplicación tutor* los cambios no se guardan en la *aplicación usuario* hasta que se realiza una sincronización**.

Desde el menú propio de este módulo que aparece en la parte superior derecha de la barra de navegación (ver fig. 64) el tutor puede gestionar para cada usuario sus tareas, su reto y sus eventos, eliminarlo o generar un informe con su progreso en formato PDF que se puede enviar por correo electrónico a cualquier destinatario, como un profesor, los padres o el propio alumno.

Al seleccionar la opción "Tareas" la vista cambia mostrando la pantalla de la fig. 65 . En la parte de arriba en el centro aparece la última puntuación que tuvo ese usuario al lado de la puntuación actual. La especificación formal del cálculo de la puntuación se expuso en el capítulo 6.4 de este documento. Para hacer la comparativa más visual se muestra una flecha roja hacia abajo en caso de que se haya empeorado o una flecha verde hacia arriba en caso de mejoría. Justo debajo aparece un listado con todas las tareas que tiene ese usuario. Estas tareas aparecen con su información más relevante (ver sección 6.2). En la esquina inferior derecha aparece el botón de añadir una nueva tarea, al presionarlo se abre una pantalla similar a la de editar una tarea (ver fig. 67) pero con todos los campos vacíos. Como ya se ha indicado anteriormente no se añadirá la nueva tarea en la *aplicación usuario* hasta que se realice una sincronización.

Figura 65: Vista de la opción de "Tareas" en un usuario

Al seleccionar una de las tareas del listado aparece una ventana de diálogo, tal y como se muestra en la fig. 66, con el nombre de la tarea y lo que se puede hacer con ella. Por un lado, se puede editar la tarea, en este caso se abre una nueva pantalla donde se puede ver el detalle de todos los campos asociados a una tarea y modificarlos tal y como se muestra en la fig. 66. Por otro lado, se permite habilitar o deshabilitar una tarea, esto significa que, aunque se sigue almacenando en la base de datos de la *aplicación tutor* no generará recordatorios ni preguntas al usuario, esta funcionalidad es de especial importancia en situaciones como las vacaciones de verano donde el profesor debe deshabilitar muchas de las tareas relacionadas con los estudios. Por último, se puede eliminar una tarea del sistema, en este caso se elimina de la base de datos del tutor instantáneamente, pero hasta que no se sincronice con el usuario no se eliminará también de la base de datos de la *aplicación usuario*.

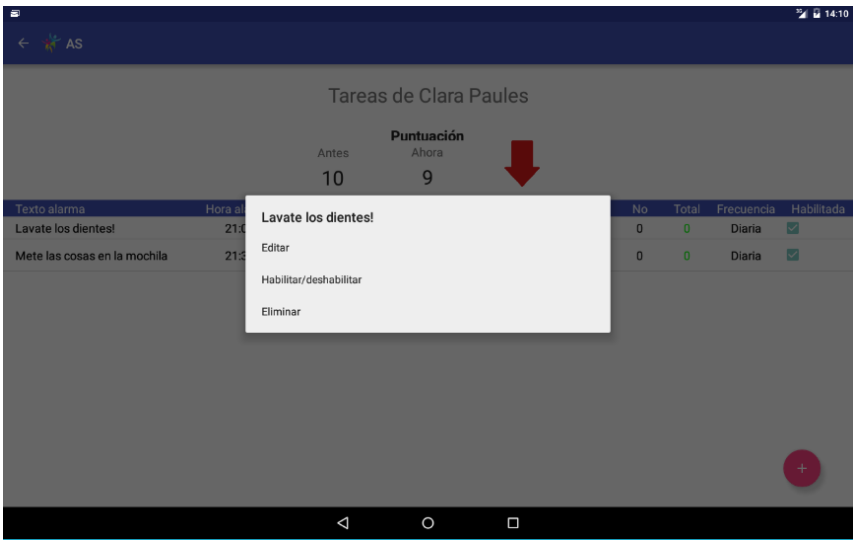


Figura 66: Diálogo de opciones sobre una tarea

Si el tutor realiza cualquier cambio en una tarea, los cambios no se actualizan en la *aplicación usuario* hasta que se haya realizado una sincronización. El único campo que no se permite editar porque se utiliza como clave para sincronizar las dos bases de datos es el texto de la alarma. En la pantalla que ilustra la fig. 67, en la parte inferior se muestran algunos datos sobre las respuestas que ha dado el usuario cuando ha sido preguntado acerca de la tarea. Estos campos tan solo se actualizan después de una sincronización con la información que ha enviado la *aplicación usuario*.

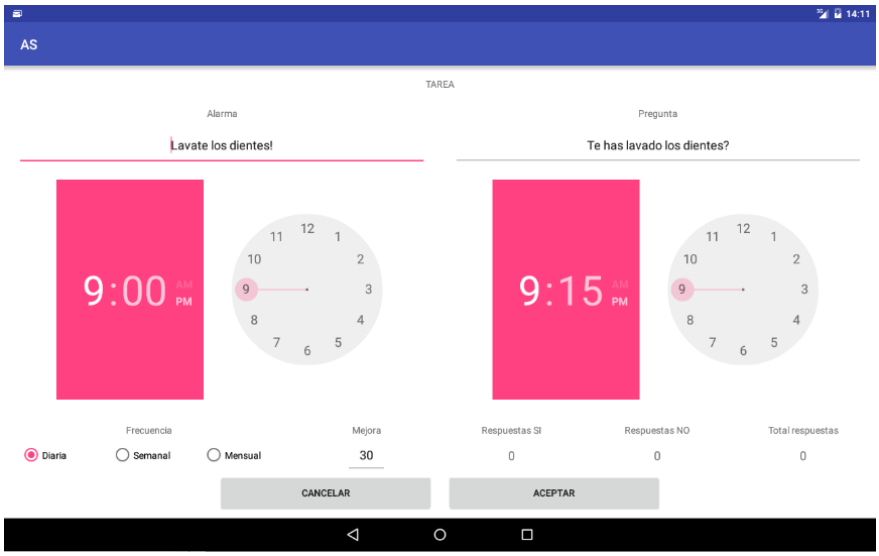


Figura 67: Vista de la creación o edición de una tarea

Cuando en el menú de usuario se selecciona la opción de “Eventos” se puede ver una lista con todos los eventos a los que ese usuario ha sido invitado y la respuesta que éste dio desde la *aplicación usuario* cuando se le preguntó por la asistencia al mismo (ver fig. 68). El valor de la asistencia se actualiza después de una sincronización entre la *aplicación tutor* y la *aplicación usuario*.

Figura 68: Vista de la opción de "Eventos" en un usuario

Si desde el menú de usuario se selecciona "Reto" hay dos posibilidades: que el usuario no tenga un reto asignado o que el usuario ya tenga un reto. (En esta primera versión de la aplicación tan solo se permite un único reto por usuario). En el primer caso, aparece la pantalla que se muestra en la fig. 69 donde el tutor puede crear un nuevo reto para el usuario introduciendo un texto con el objetivo del desafío y otro con el premio que conseguirá el alumno si llega a conseguir realizarlo treinta veces.

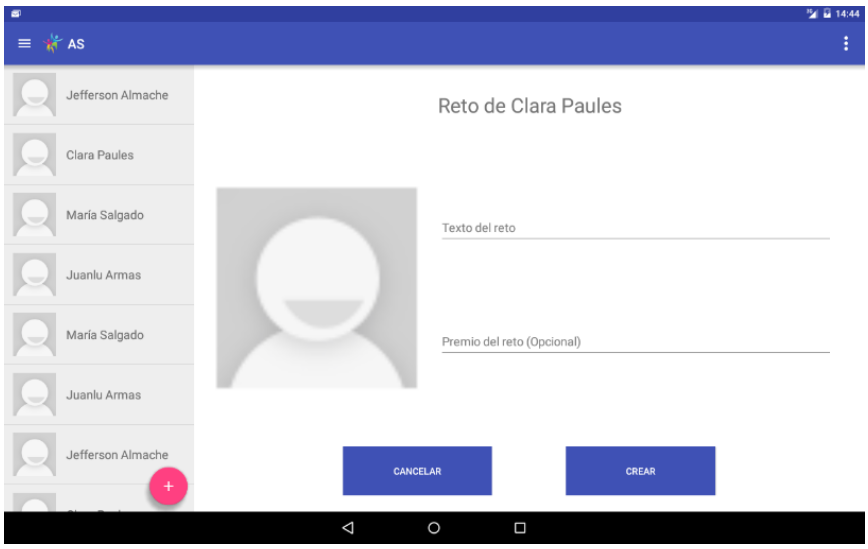


Figura 69: Vista de la opción de "Reto" de un usuario cuando no tenía ninguno

En el segundo caso, aparece una pantalla donde se muestra la información del reto y además una barra de progreso con el número de veces que el usuario lo ha superado de momento. (Ver fig. 70).

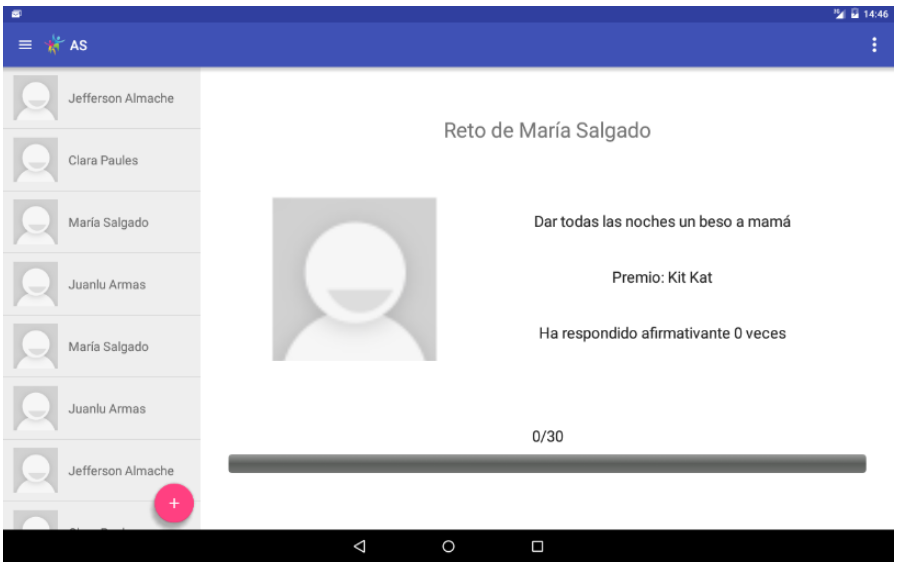


Figura 70: Vista de la opción de "Reto" de un usuario si hay uno existente

La cuarta opción del menú de usuario es "Enviar informe". Si se selecciona esta opción se abre la aplicación de correo predeterminada del dispositivo y se muestra un correo electrónico con un PDF generado en ese momento que contiene información útil para seguir el progreso del usuario: su puntuación, sus tareas, sus respuestas y su reto. Se pueden editar los destinatarios y siempre aparecerá como remitente el correo que se introdujera en el momento de registrarse en la aplicación.

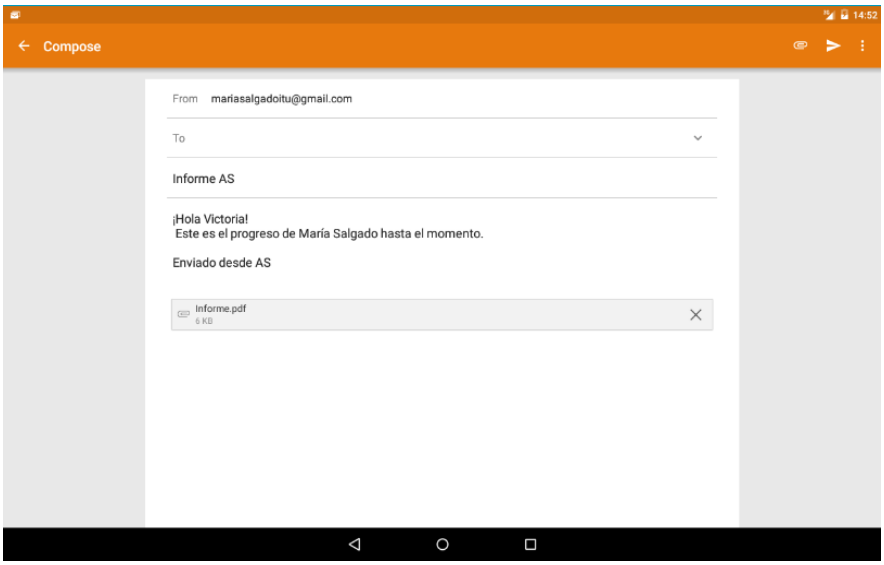
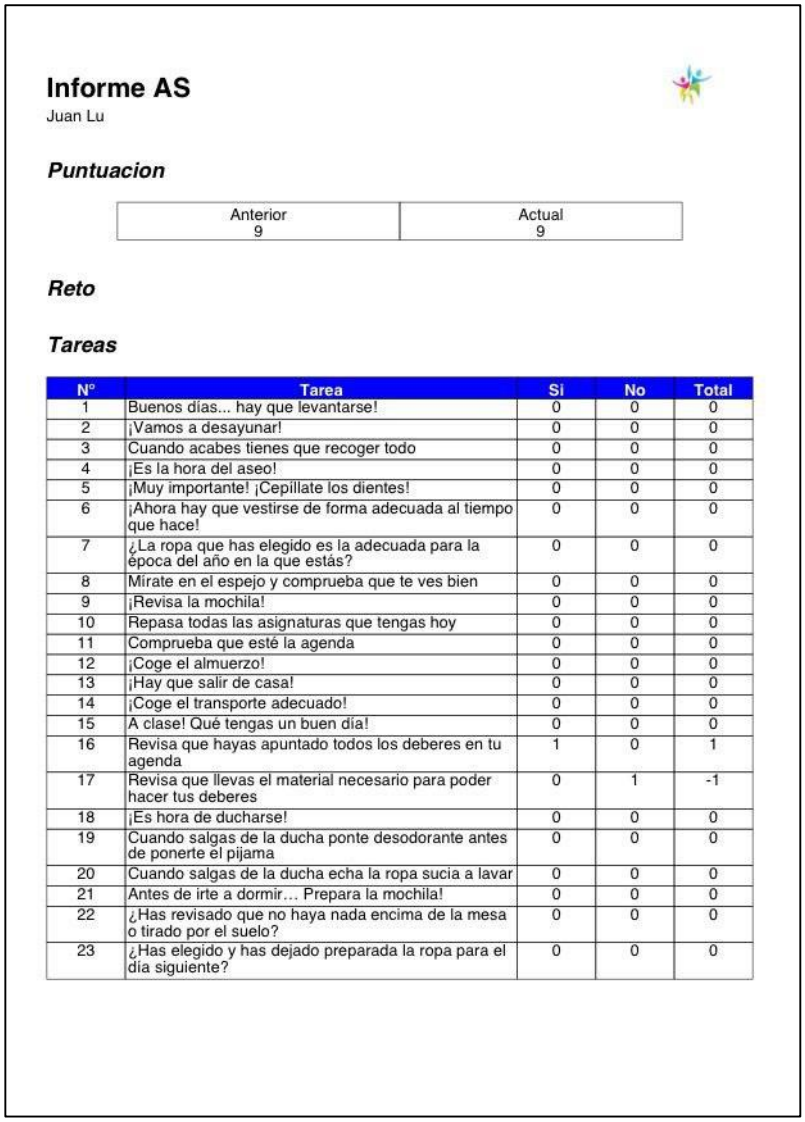


Figura 71: Correo que se envía desde AS con el informe del usuario



Informe AS

Juan Lu

Puntuacion

Anterior	Actual
9	9

Reto

Tareas

Nº	Tarea	Si	No	Total
1	Buenos días... hay que levantarse!	0	0	0
2	¡Vamos a desayunar!	0	0	0
3	Cuando acabes tienes que recoger todo	0	0	0
4	¡Es la hora del aseo!	0	0	0
5	¡Muy importante! ¡Cepillate los dientes!	0	0	0
6	¡Ahora hay que vestirse de forma adecuada al tiempo que hace!	0	0	0
7	¿La ropa que has elegido es la adecuada para la época del año en la que estás?	0	0	0
8	Mírate en el espejo y comprueba que te ves bien	0	0	0
9	¡Revisa la mochila!	0	0	0
10	Repasa todas las asignaturas que tengas hoy	0	0	0
11	Comprueba que esté la agenda	0	0	0
12	¡Coge el almuerzo!	0	0	0
13	¡Hay que salir de casa!	0	0	0
14	¡Coge el transporte adecuado!	0	0	0
15	A clase! Qué tengas un buen día!	0	0	0
16	Revisa que hayas apuntado todos los deberes en tu agenda	1	0	1
17	Revisa que llevas el material necesario para poder hacer tus deberes	0	1	-1
18	¡Es hora de ducharse!	0	0	0
19	Cuando salgas de la ducha ponte desodorante antes de ponerte el pijama	0	0	0
20	Cuando salgas de la ducha echa la ropa sucia a lavar	0	0	0
21	Antes de irte a dormir… Prepara la mochila!	0	0	0
22	¿Has revisado que no haya nada encima de la mesa o tirado por el suelo?	0	0	0
23	¿Has elegido y has dejado preparada la ropa para el día siguiente?	0	0	0

Figura 72: Informe generado por la aplicación que se adjunta en el correo

El segundo módulo del menú principal (ver fig. 62) es **"Eventos"**. Este módulo permite crear, editar y eliminar eventos a los que se puede invitar a los usuarios del sistema. Cuando se selecciona esta opción aparece la pantalla que se muestra en la fig. 73. En la parte izquierda se muestra un listado con los eventos existentes y al seleccionar uno de ellos en la parte central aparecen todos los detalles de dicho evento.

Para crear un evento, el tutor debe seleccionar el icono de añadir  , en ese momento aparecerá pantalla desde donde completar la información de un evento: el título, la hora de inicio, la hora de notificar un recordatorio, la fecha y la lista de invitados. Este último elemento es muy interesante porque permite consultar quiénes han confirmado su asistencia. Como ya se ha insistido anteriormente, cualquier cambio se actualizará después de la sincronización entre la *aplicación usuario* y la *aplicación tutor*.

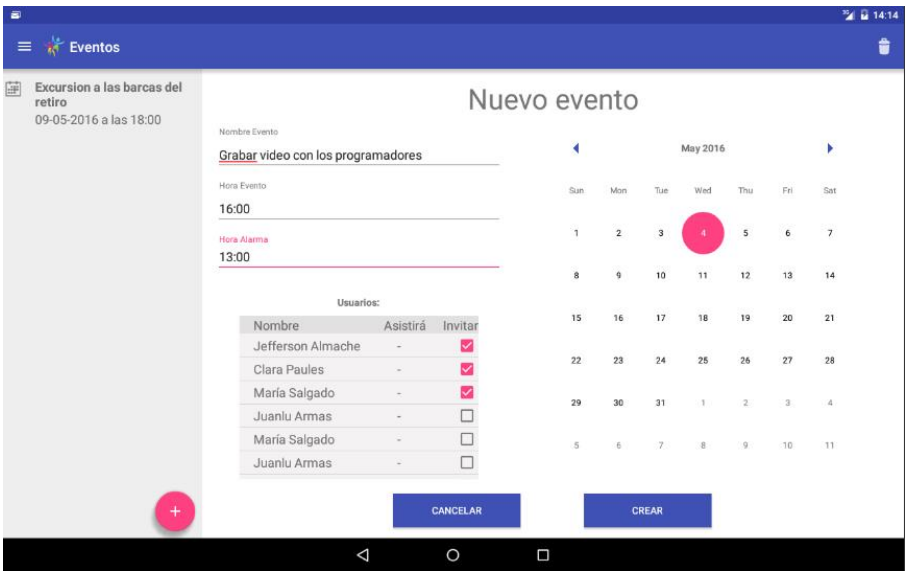


Figura 73: Creación de un evento

El tercer módulo del menú principal (ver fig. 62) es **"Mi perfil"**. Éste permite configurar los datos del propio tutor. Al seleccionar esta opción aparece la pantalla que se muestra en la fig. 74 desde donde se puede editar el nombre, el correo, la pregunta de seguridad, la respuesta a la pregunta de seguridad y la contraseña que protege la aplicación.

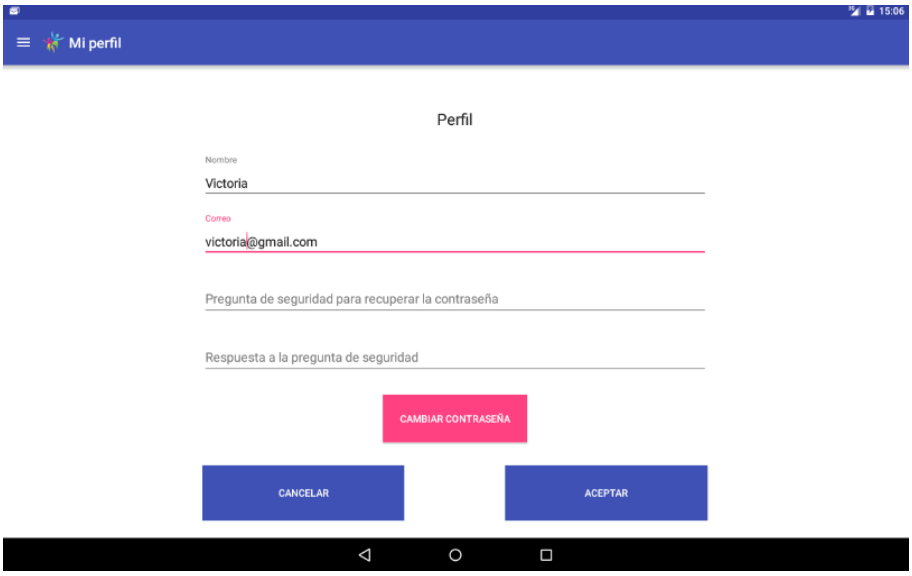


Figura 74: Edición de "Mi perfil" en la *aplicación tutor*

La contraseña es un campo obligatorio debido al carácter privado y de acceso restringido de esta aplicación. Cada vez que se abre la aplicación se solicita una clave de acceso como se muestra en la fig. 75. La aplicación tan solo se inicia y muestra su menú principal en caso de

introducir la contraseña correctamente. En caso de haber olvidado la contraseña y querer recuperarla se debe responder correctamente a la pregunta de seguridad que se almacenó en base de datos al crear registrarse. Esta funcionalidad se añadió debido a que la tablet en la que se instaló la *aplicación tutor* para la realización de pruebas es de acceso público para cualquier alumno que la necesite. Por este motivo se vio necesario bloquear la aplicación de manera que solo personas autorizadas que conozcan la clave puedan ejecutarla.

Figura 75: Clave de acceso para desbloquear la *aplicación tutor*

El último módulo del menú principal (ver fig. 62) es **"Ayuda"**. Al seleccionar esta opción aparece un listado con las preguntas en modo FAQs que se han considerado más importantes y al seleccionar una de las preguntas se despliega su respuesta acompañada con un pantallazo propio tal y como se muestra en la fig. 76.

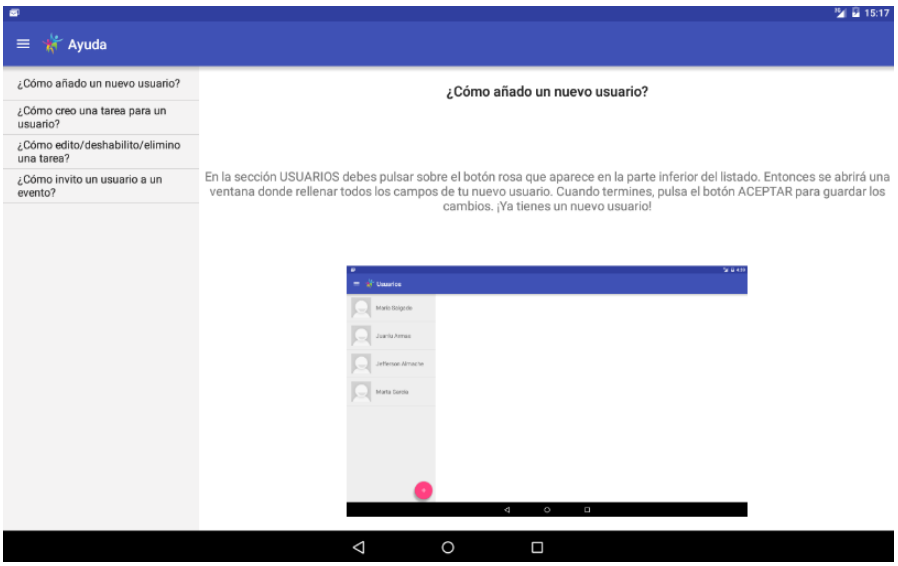


Figura 76: Ayuda de la *aplicación tutor*

7.3 Base de datos

Los datos se almacenarán en una base de datos local en la propia tablet lo que hace su acceso rápido y sin necesidad de una conexión a Internet.

En la fig. 77 se puede ver el diagrama que ilustra el modelo entidad relación con el que se ha diseñado la base de datos. Se destacan los atributos que son clave primaria en cada tabla. Hay seis tablas que almacenan la información propia de los distintos objetos de negocio:

- *Tutor:* tiene la información propia del perfil del profesor que maneja la *aplicación tutor*. Esta tabla solo tiene una fila porque solo hay un profesor en un sistema AS.

- *Usuario:* contiene todos los alumnos que el profesor tiene dados de alta en la *aplicación tutor*. Cada fila contiene la información de un alumno. Esta tabla tiene una relación 1 a N con la tabla "Tarea" , 1 a 1 con la tabla "Reto" y N a M con la tabla "Evento". Esta última genera una nueva tabla, *UsuarioEvento*.

- *Tarea:* contiene todas las tareas existentes en la *aplicación tutor*. Cada fila es la información de un suceso tipo tarea relacionándola con el usuario al que es adjudicada. Un usuario puede tener muchas tareas pero una tarea solo pertenece a un usuario (Relación N, 1).

- *Reto:* contiene todos los retos existentes en la *aplicación tutor*. Cada fila respresenta la información de un suceso tipo reto relacionándolo con el usuario al que es adjudicado. Un usuario solo puede tener un reto y viceversa (Relación 1, 1).

- *Evento:* contiene todos los eventos existentes en la *aplicación tutor*. Cada fila se corresponde con la información de un suceso tipo evento, no almacena los usuario invitados. Un usuario puede estar invitado a muchos eventos y un evento tiene invitados a muchos usuarios (Relación N, M). Se genera la tabla *UsuarioEvento*.

- *UsuarioEvento:* cada fila relaciona cada evento con cada usuario invitado. Esto se debe a que un usuario puede estar invitado a muchos eventos y a que un evento puede tener invitados a muchos usuarios. (Relación N a M)

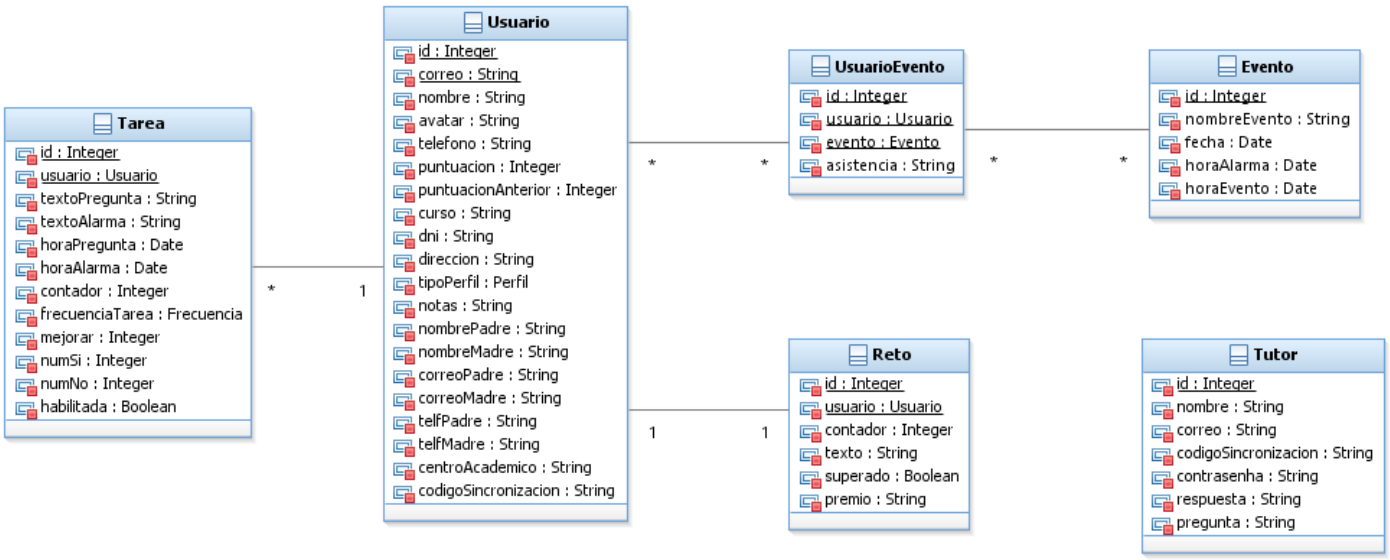


Figura 77: Diagrama entidad relación de la *aplicación tutor*

CAPÍTULO 8: Comunicación entre los dos tipos de aplicación

"La única manera de encontrar
los límites de lo posible
es ir más allá de lo imposible."
ARTHUR C. CLARKE

En este capítulo se explica cómo se intercambia la información entre la *aplicación tutor* y la *aplicación usuario*. Esta parte del sistema, destacada en color en la fig. 78, se considera vital debido a que el objetivo de AS es que los profesionales que trabajan con personas Asperger puedan gestionar la información de cada alumno desde la *aplicación tutor*, instalada en el dispositivo del profesor. Todos los cambios realizados en un usuario desde la *aplicación tutor* se actualizarán en la *aplicación usuario* después de realizar una sincronización. De esta manera se evita que los profesores tengan que interactuar con los dispositivos de cada alumno y además se permite que puedan tener centralizada la información de todos en un solo dispositivo.

Comenzaremos exponiendo las situaciones en las que es necesaria la comunicación entre los dos tipos de aplicación para un correcto funcionamiento del sistema, después definiremos la arquitectura de la comunicación, a continuación, explicaremos cómo se realiza el intercambio de información y por último plantearemos un caso ejemplo de sincronización.

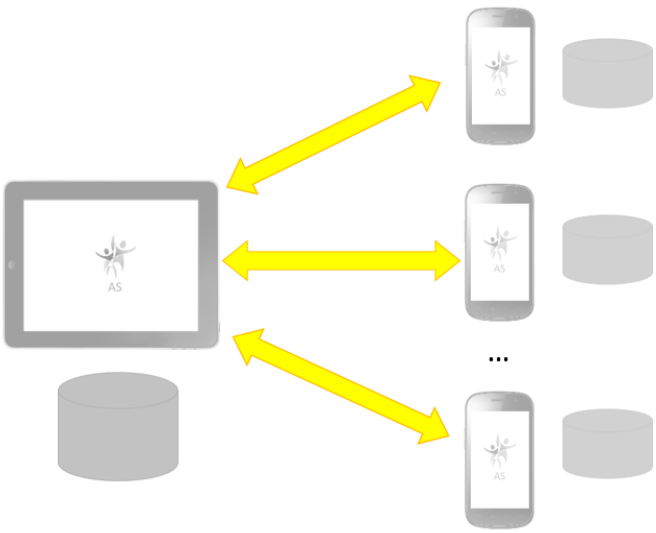


Figura 78: Esquema de la arquitectura global del sistema (conexiones)

8.1 Situaciones en las que se recomienda la comunicación

Las dos aplicaciones del sistema AS tienen un funcionamiento normal independiente entre sí. Al utilizar cada una de ellas se realizan modificaciones en sus propias bases de datos locales. Al realizar una sincronización se combinan los datos de ambas aplicaciones para obtener una información actualizada (ver fig. 79). En esta combinación prevalece siempre la información existente en la BBDD de la *aplicación tutor* porque la *aplicación usuario* no tiene posibilidad de editar tareas, eventos o retos. De la *aplicación usuario* tan solo se mantienen los contadores para un correcto seguimiento del progreso, es decir, el número de respuestas afirmativas o negativas, la intención de asistencia a eventos y la puntuación. Además existen datos que son propios de cada aplicación y no se intercambian: los nombres de los usuarios y los avatares.

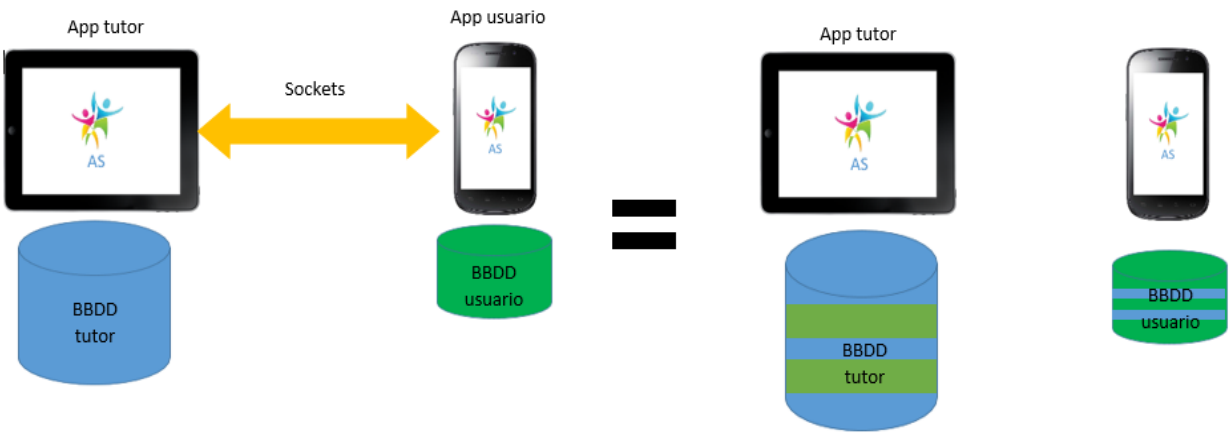


Figura 79: Combinación de información en BBDD tras la sincronización

Por este motivo, se recomienda realizar una sincronización siempre que se quiera tener la información actualizada en las dos aplicaciones. Algunos casos importantes son:

- Para registrarse en la *aplicación usuario* es necesario que se efectúe una primera comunicación entre las dos aplicaciones para evitar que la *aplicación usuario* tenga una base de datos vacía. Por este motivo el tutor debe previamente dar de alta al nuevo usuario en la *aplicación tutor*.
- Cuando en la *aplicación tutor* se añade algún suceso a un usuario. Hasta que no se realice una sincronización la *aplicación usuario* no generará recordatorios ni preguntas de ese suceso.
- Cuando en la *aplicación tutor* se realiza alguna modificación en un suceso existente. Mientras no se realice la sincronización el cambio no se efectuará en la *aplicación usuario*.

- Cuando el tutor quiera consultar la información de progreso actualizada de un usuario. Al comunicarse la *aplicación usuario* envía el número de respuestas afirmativas y negativas de las tareas, el contador de su reto y la asistencia a los eventos. Estos valores se actualizan en la *aplicación tutor* después de la sincronización y entonces sí se puede disponer de la información actualizada.

8.2 Arquitectura del sistema de comunicación

Para implementar el sistema de comunicación de AS se valoraron principalmente dos tecnologías de comunicación: a través de un servidor o mediante sockets. Finalmente se resolvió el dilema haciendo uso de *sockets* [11]. La principal razón para no incluir ningún servidor o intermediario en AS es porque queremos que el sistema pueda ser siempre utilizado de manera gratuita. Si se utilizaran servidores habría que costear los gastos de mantenimiento y, además, al introducir un agente externo en el sistema, existirían problemas de privacidad debido a que se trata de información considerada sensible por la Ley Orgánica de Protección de Datos (menores y discapacitados). Con el uso de sockets estos problemas no existen ya que los datos se guardan en bases de datos locales de cada dispositivo y los profesores especialistas que tratan con enfermos Asperger tienen todos los permisos para disponer de su información.

Un socket es un concepto abstracto por el cual dos programas intercambian cualquier flujo de datos de manera fiable y ordenada [12]. Tal y como se muestra en la fig. 80 se utiliza la arquitectura cliente-servidor. En nuestro sistema la *aplicación tutor* actúa como servidor y la *aplicación usuario* actúa como cliente. Es importante especificar que para hacer uso de sockets ambas aplicaciones deben encontrarse en la misma red WiFi debido a que se hace uso de esta red. Es necesario que el cliente y el servidor sean capaces de localizarse el uno al otro a través de las direcciones IP de los dispositivos.

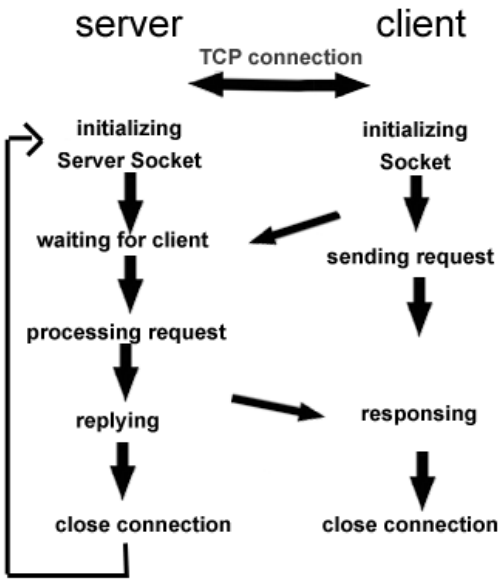


Figura 80: Diagrama de la arquitectura cliente-servidor mediante sockets

El uso de sockets ha supuesto un reto tecnológico para los desarrolladores ya que en ninguna asignatura del Grado de Ingeniería del Software que finalizamos con este proyecto hemos estudiado su funcionamiento ni los hemos usado para ninguna práctica.

8.3 Intercambio de información

Es relativamente fácil para un desarrollador medio conseguir que dos programas intercambien tipos de datos primitivos como cadenas de caracteres o enteros pero en el sistema AS lo interesante es el intercambio de objetos. En concreto, debido al uso de patrones de ingeniería del software basada en componentes, explicados en el capítulo 5, los objetos que se envían son *Data Transfer Object*(DTO) que representan los *Data Access Object*(DAO). Para conseguir el envío y recepción de objetos, estos deben ser serializables [13] lo cual permite codificar sus atributos a nivel de bits para descomponer los objetos en el envío y poder reconstruirlos al terminar la conexión.

La información que se intercambian los dos tipos de aplicación está almacenada en unos objetos de tipo *Mensaje*. Esta clase solamente tiene *getters* y *setters* y es la única clase idéntica en ambas aplicaciones ya que, aunque tienen el mismo nombre, los objetos de negocio no son exactamente iguales en las dos aplicaciones porque la información que se necesita no es la misma. Por ejemplo, la *aplicación usuario* no necesita saber todos los invitados a un evento pero en la *aplicación tutor* ésta información es fundamental, por este motivo el objeto *Evento* en la *aplicación usuario* no guardará estos datos y sí los guarda en la *aplicación tutor*.

En la comunicación de nuestro sistema un mensaje está formado por:

- Una cadena llamada *verificar* que es la palabra clave para identificar cómo procesar un mensaje. Toma valores como “Registro”, “Permitido” o “Cod:”.
- Un *TransferUsuario* que contiene toda la información del usuario.
- Un *TransferReto* que representa el reto del usuario.
- Una lista de *TransferTarea* que contiene todos los datos de las distintas tareas que tiene el usuario.
- Una lista de *TransferEvento* que contiene toda la información de los eventos a los que está invitado el usuario.

La *aplicación tutor* es capaz de procesar el mensaje que le llega del usuario. Esto quiere decir que tan solo consulta los campos que interesan como los contadores de las tareas para saber cuántas respuestas afirmativas y negativas ha realizado y almacena esta información. El mensaje que crea con la nueva información es una combinación de todo lo que el tutor tenía en base de datos y lo que acaba de almacenar del mensaje del usuario. Esta mezcla es muy importante porque evita que se pierda información importante del usuario como los contadores de las tareas o de progreso de un reto.

La *aplicación usuario* simplemente crea objetos, forma mensajes y los envía o recibe mensajes y los guarda sobrescribiendo la información que tenía guardada. Por el contrario, la *aplicación tutor* es capaz de procesar el mensaje distinguiendo para cada objeto en qué campos debe prevalecer su información actual y qué campos debe sobrescribir con los datos del mensaje que le llega. Precisamente por este motivo primero lee el mensaje que le envía el usuario, lo procesa mezclando la información que tenía el tutor almacenada con la información nueva que le envía el usuario y después envía a la *aplicación usuario* un nuevo objeto mensaje con los datos definitivos obtenidos a partir de la mezcla.

8.4 Planteamiento de un caso de sincronización

En este apartado vamos a explicar el flujo de información entre ambas aplicaciones mediante el planteamiento de un caso ejemplo: el tutor ha añadido dos nuevas tareas y un reto a un usuario en su casa por la noche, al día siguiente en la sede de la asociación, cuando ya ha llegado el usuario, quiere sincronizar con él para que se actualicen sus datos con las modificaciones.

Antes de explicar el flujo hay que destacar:

- ➢ Que los dispositivos donde están instaladas ambas aplicaciones deben estar conectadas a la misma red WiFi. Si alguna de ellas está usando una conexión propia 3G o de cualquier otro tipo la sincronización no tendrá éxito.
- ➢ Que cada usuario tiene un código único que se generó cuando el tutor lo creó en la *aplicación tutor* y con el cual el usuario se registró cuando se instaló la *aplicación usuario*. En nuestro ejemplo el código de sincronización del usuario es VIC01.

A continuación, se expone el flujo que además se ilustra en el diagrama de la fig. 85 relacionando las partes de la imagen con la numeración de esta explicación:

1. El tutor selecciona el usuario desde la *aplicación tutor* y presiona el botón de sincronización inicializando el socket servidor.
2. El usuario, desde la *aplicación usuario* también debe seleccionar el botón de sincronización para comenzar a recorrer todas las direcciones IP de esa red WiFi (Desde la 192.168.0.1 a la 192.168.0.255) hasta encontrar la IP del dispositivo con la *aplicación tutor*, entonces envía una petición con un mensaje en el que indica su código de sincronización tal y como se muestra en la fig. 81.

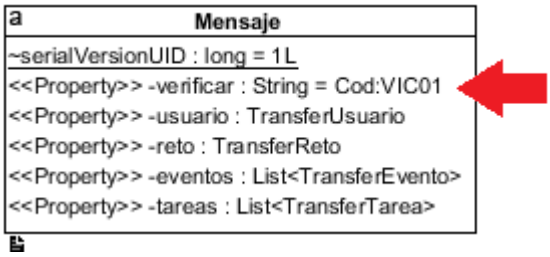


Figura 81: Mensaje generado por la *aplicación usuario* indicando su código de sincronización

3. La *aplicación tutor* responde a la petición del usuario enviando un mensaje con el texto "Permitido" (ver fig. 82) porque el código es el del usuario que está esperando (VIC01). En caso contrario ignora la petición y continúa esperando. Esto impide que otros alumnos que estén intentando sincronizar a la vez con el tutor corrompan la sincronización.

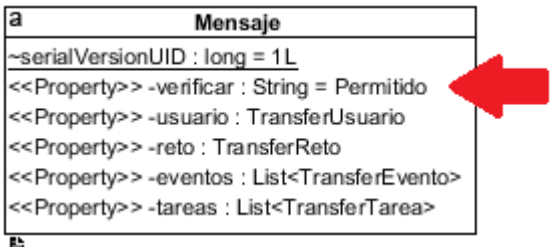


Figura 82: Mensaje generado por la *aplicación tutor* "Permitido"

4. Como a la *aplicación usuario* detecta que le ha llegado el mensaje respuesta "Permitido", hace una búsqueda de toda la información disponible en la BBDD local, crea un mensaje con esta información como se ilustra en la fig. 83 y se lo envía al tutor.

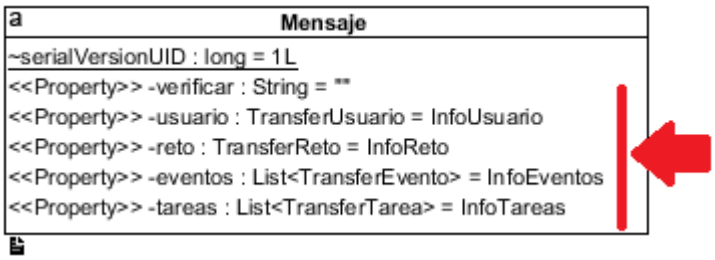


Figura 83: Mensaje generado por la *aplicación usuario* con toda su información

5. La *aplicación tutor* recibe el mensaje, lo procesa y genera un mensaje con toda la información de ese usuario actualizada (Ver fig. 84), se lo envía al usuario y finaliza la conexión cerrando el socket servidor. En ese mensaje incluye la nueva información de las tareas modificadas y el nuevo reto para el usuario en este caso.

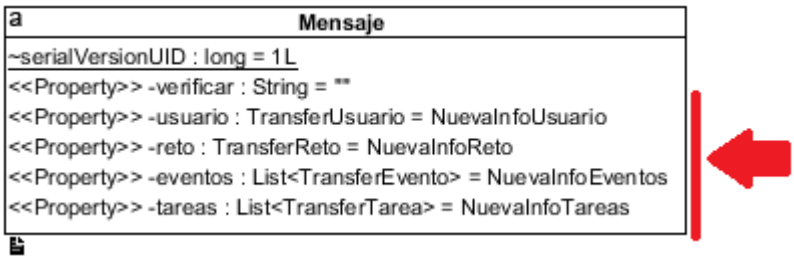


Figura 84: Mensaje generado por la *aplicación tutor* con la nueva información

6. El usuario recibe ese mensaje, almacena la nueva información y también termina la conexión cerrando su socket. Al guardar esta información sobrescribe los datos que tuviera, es decir, la información antigua se elimina quedando solamente la nueva información que el tutor ha enviado a través del mensaje.

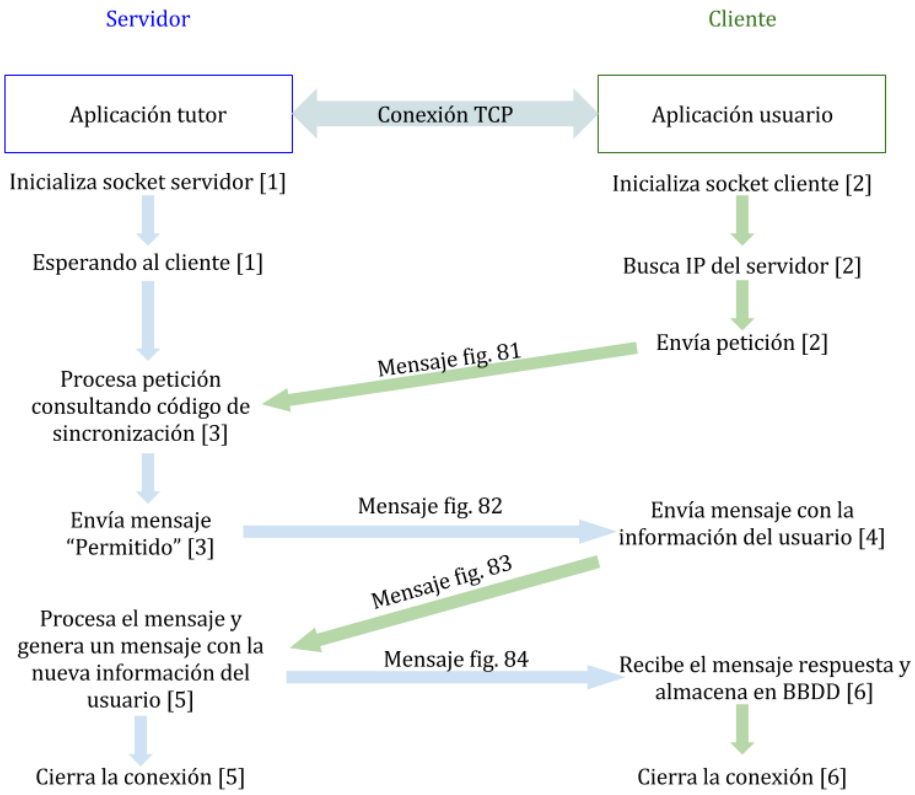


Figura 85: Diagrama de un flujo de sincronización exitosa entre los dos tipos de aplicación

Cualquier otro caso en el que se sincronizan las dos aplicaciones es análogo a este ejemplo salvando cuando un usuario se está registrando en la *aplicación usuario*. La diferencia es que el primer mensaje que envía el usuario tiene todos los campos nulos dado que aún no se tiene información en BBDD. Se da valor "Registro" al campo *verificar* del mensaje para que la *aplicación tutor* distinga este caso, se salte el procesamiento del mensaje y directamente envíe la información que tiene en la BBDD para que se cargue en la *aplicación usuario*.

CAPÍTULO 9: Casos de uso

"Los programadores hablan sobre desarrollo
los fines de semana, vacaciones y en las comidas,
no por falta de imaginación, sino porque su imaginación
revela mundos que otros no pueden ver."
LARRY O'BRIEN y BRUCE ECKEL en "Thinking in C#"

9.1 Aplicación usuario

- Editar perfil

Función	EDITAR PERFIL
Prioridad	Alta
Estabilidad	Alta
Descripción	Modifica los datos del usuario
Entrada	Datos a modificar
Salida	Mensaje
Origen	Operador del sistema usuario.
Destino	Sistema.
Precondición	-
Postcondición	Se guardan las modificaciones de los datos del usuario en el sistema
Flujo de evento	**1.** El usuario pulsa sobre el botón de configuración de la aplicación. **2.** La pantalla ofrece las diversas opciones que el usuario puede elegir modificar (avatar, color, nombre y sonido de las alertas). Si selecciona la imagen le aparece un submenú para abrir la cámara del dispositivo y hacer una foto o elegir de la galería o dejar la imagen por defecto. **3.** El usuario pulsa sobre aceptar: a. Si no se modifica nada: no se actualiza el sistema, vuelve a la pantalla principal del usuario. b. Si se han cambiado los datos: Se actualiza la información en el sistema y se pasa a la pantalla de principal del usuario. Se muestra un mensaje de éxito.
Efectos laterales	-

- Mostrar ayuda

Función	MOSTRAR AYUDA
Prioridad	Media
Estabilidad	Alta
Descripción	Muestra una guia de usuario de la *aplicación usuario* para la pantalla en la que se encuentra.
Entrada	-
Salida	Vista de la ayuda
Origen	Operador del sistema usuario.
Destino	Sistema.
Precondición	-
Postcondición	-
Flujo de evento	**1.** En la parte superior derecha de la aplicación, el usuario selecciona el icono de ayuda. **2.** La aplicación muestra una ventana que contiene una imagen con un pantallazo de la vista de la pantalla en la que se encuentra con algunas aclaraciones y sobre la que es posible hacer zoom.
Efectos laterales	-

- Ver informe

Función	VER INFORME
Prioridad	Alta
Estabilidad	Alta
Descripción	Muestra la evolución del usuario a través de una comparativa de puntuaciones y una lista de las tareas que el sistema le recuerda.
Entrada	-
Salida	Pantalla descriptiva con la última información del usuario
Origen	Operador del sistema usuario.
Destino	Sistema.
Precondición	-
Postcondición	-
Flujo de evento	**1.** El usuario selecciona en la pantalla principal la opción de "¿Cómo vas?" **2.** Aparece una nueva pantalla donde el usuario puede ver una comparativa con su última puntuación y la actual para visualizar su progreso general (se ilustra con una flecha hacia arriba si se ha mejorado y una flecha hacia abajo si la puntuación es peor). Debajo se muestra una tabla donde aparece cada tarea con el número de veces que el usuario respondió "SI", el número de veces que el usuario respondió "NO" y el balance entre el número de respuestas afirmativas y negativas (verde si es positivo, rojo sino).* (*) El usuario no puede modificar ninguna tarea ni realizar ninguna acción con las mismas.
Efectos laterales	-

- Ver eventos

Función	VER EVENTOS
Prioridad	Alta
Estabilidad	Alta
Descripción	Se muestra la lista de eventos a los que el usuario ha sido invitado por el tutor con un checkbox en el que puede indicar su intención de asistir.
Entrada	-
Salida	-
Origen	Operador del sistema usuario.
Destino	Sistema.
Precondición	-
Postcondición	-
Flujo de evento	**1.** El usuario desde la pantalla principal selecciona la opción "Ver eventos". **2.** La interfaz cambia mostrando una nueva ventana con un listado de los eventos. Aparece el nombre de la actividad, la fecha, la hora y una columna con checkbox para que el usuario pueda indicar su intención de asistir. Si no tuviera eventos se mostraría un mensaje "No tienes ningún evento en este momento".
Efectos laterales	-

- Asistir/No asistir a un evento

Función	ASISTIR/NO ASISTIR A UN EVENTO
Prioridad	Alta
Estabilidad	Alta
Descripción	El usuario a través de un checkbox indica su intención de asistir a un evento.
Entrada	ID Evento
Salida	-
Origen	Operador del sistema usuario.
Destino	Sistema.
Precondición	-
Postcondición	-
Flujo de evento	**1.** El usuario desde la pantalla “Ver eventos” selecciona la fila en la que se le muestra la información de un evento para marcar el checkbox en caso de que quiera asistir o desmarcarlo en caso contrario. **2.** Si el usuario presiona después el botón “GUARDAR” los cambios se guardarán para transmitirlos a la *aplicación tutor* en la siguiente sincronización y se regresa a la pantalla principal.
Efectos laterales	-

- Ver reto

Función	VER RETO
Prioridad	Alta
Estabilidad	Alta
Descripción	Muestra el reto del usuario con su información y su progreso.
Entrada	-
Salida	-
Origen	Operador del sistema usuario.
Destino	Sistema.
Precondición	-
Postcondición	-
Flujo de evento	**1.** El usuario selecciona desde su pantalla principal la opción "Ver reto". **2.** La interfaz cambia a una nueva pantalla en la que aparece el título del reto en la parte superior, el premio que conseguirá el usuario si lo realiza, dos botones grandes con las opciones "SI" o "NO" y bajo ellos aparece una barra de progreso con el número de veces que se haya cumplido el reto. Si no tuviera un reto se mostraría un mensaje "No tienes ningún reto en este momento".
Efectos laterales	-

- Responder reto

Función	RESPONDER RETO
Prioridad	Alta
Estabilidad	Alta
Descripción	El usuario indica si ha realizado o no la actividad que se le indicaba en el reto.
Entrada	ID reto
Salida	Mensaje
Origen	Operador del sistema usuario.
Destino	Sistema.
Precondición	Reto existente
Postcondición	-
Flujo de evento	**1.** En la pantalla de "Retos", el usuario pulsa sobre el botón que corresponda SI (+1) o NO (-1) según haya realizado la actividad indicada o no. Se confía en la sinceridad del usuario. **2.** Se suma la respuesta al contador de progreso en BBDD y se actualiza la barra de progreso.
Efectos laterales	-

- Enviar informe por correo

Función	ENVIAR INFORME POR CORREO
Prioridad	Alta
Estabilidad	Alta
Descripción	Se envía un informe con la información del usuario en formato PDF adjunta.
Entrada	-
Salida	Correo electrónico e informe en PDF.
Origen	Operador del sistema usuario.
Destino	Sistema
Precondición	-
Postcondición	-
Flujo de evento	**1.** El usuario desde la pantalla principal selecciona la opción “Enviar informe por correo”. **2.** Se genera un documento PDF con datos del usuario como los sucesos que tiene en ese momento y una comparativa de la puntuación. Se almacena en la carpeta de descargas en un directorio que se llama AS. Se sobreescribe cada vez, no se almacena. **3.** Se pasa la aplicación AS en segundo plano y se abre la aplicación de correo electrónico que el usuario tenga instalada y activa en su móvil. Aparece el correo del usuario como destinatario predefinido y un pequeño texto de saludo y explicación en el correo además del informe en PDF adjunto.
Efectos laterales	-

- Mostrar alarma

Función	MOSTRAR ALARMA
Prioridad	Alta
Estabilidad	Alta
Descripción	Muestra una notificación para recordar la realización de una tarea.
Entrada	ID tarea
Salida	Notificación
Origen	Operador del sistema usuario.
Destino	Sistema.
Precondición	Tarea existente
Postcondición	-
Flujo de evento	**1.** Cuando la hora actual coincide con la hora de la alarma de una tarea, la aplicación lanza una notificación con el texto de alarma de la tarea. **2.** El usuario la descarta desplazando o con el gesto propio de su dispositivo.
Efectos laterales	-

- Responder pregunta

Función	RESPONDER PREGUNTA
Prioridad	Alta
Estabilidad	Alta
Descripción	El usuario responde afirmativamente o negativamente si ha realizado una tarea.
Entrada	ID tarea
Salida	Notificación
Origen	Operador del sistema usuario.
Destino	Sistema.
Precondición	Tarea existente
Postcondición	
Flujo de evento	**1.** Cuando la hora actual coincide con la hora la pregunta de una tarea, la aplicación lanza una notificación con el texto de pregunta de la tarea. **2.** El usuario puede contestar: a. SI: se añade un acierto al contador de progreso y se ejecuta el algoritmo que inteligentemente cambia la frecuencia si es preciso. b. NO: se resta un acierto al contador de progreso y se ejecuta el algoritmo que cambia la frecuencia si es necesario. Si se detectan 3 respuestas negativas seguidas se aumenta la frecuencia de nuevo. c. El usuario no contesta descartando la notificación: no se realiza ningún cambio en los contadores de respuesta. **3.** La respuesta se almacena en el sistema usuario y desaparece la notificación.
Efectos laterales	-

- Sincronizar

Función	SINCRONIZAR
Prioridad	Alta
Estabilidad	Alta
Descripción	Sincroniza la aplicación del tutor con la aplicación del usuario intercambiando la información necesaria y pertinente para cada tipo de aplicación.
Entrada	-
Salida	Mensaje
Origen	Operador del sistema usuario
Destino	Sistema
Precondición	Usuario dado de alto en la *aplicación tutor*. Sólo se ejecutará correctamente la conexión si los dos dispositivos (tutor y usuario) se encuentran conectados a la misma red WiFi. Se debe haber seleccionado la opción de sincronizar desde la *aplicación tutor* con ese usuario primero.
Postcondición	-
Flujo de evento	**1.** El usuario selecciona el botón de sincronización que se encuentra en la pantalla principal. Se envía una petición de sincronización a la *aplicación tutor*. **2.** Se muestra mensaje "Sincronizando...". Y si la *aplicación tutor* acepta la petición se continúa, sino se continúa con el paso 5. **3.** Se guarda toda la información actual del usuario en un mensaje que se envía a la *aplicación tutor*. **4.** Se recibe el mensaje respuesta y se almacena en BBDD la nueva información sobrescribiendo la antigua. **5.** Se muestra mensaje de éxito o error según el desarrollo de la sincronización: a. "Sincronización correcta" si todo se ha desarrollado normalmente. b. "Error en la sincronización" seguido de la descripción de qué ocasionó el fallo. **6.** Se cierra la conexión.
Efectos laterales	-

- Registro

Función	REGISTRO
Prioridad	Alta
Estabilidad	Alta
Descripción	Sincroniza la aplicación del tutor con la aplicación del usuario por primera vez cargando la información que se envía y la recogida en el formulario de registro en BBDD.
Entrada	Datos del formulario de registro.
Salida	Mensaje
Origen	Operador del sistema usuario.
Destino	Sistema
Precondición	Usuario dado de alto en la *aplicación tutor*. Sólo se ejecutará correctamente la conexión si los dos dispositivos (tutor y usuario) se encuentran conectados a la misma red WiFi. Se debe haber seleccionado la opción de sincronizar desde la *aplicación tutor* con ese usuario primero.
Postcondición	-
Flujo de evento	**1.** El usuario abre por primera vez la aplicación en su móvil. **2.** Aparece una pantalla de registro en la que se solicita: nombre, correo y clave del tutor. La clave del tutor es un código alfanumérico que el tutor debe indicarle. **3.** Tras rellenar los datos selecciona "ACEPTAR". **4.** Se realiza la primera sincronización con el dispositivo del tutor y se carga toda la información necesaria en la *aplicación usuario* (Ver caso de uso "Sincronizar"). **5.** Tras la sincronización: a. En caso de éxito se muestra la pantalla principal de la aplicación. b. En caso de error se continúa en la pantalla de registro.
Efectos laterales	-

9.2 Aplicación tutor

9.2.1 Casos de uso relacionados con el tutor

- Registro

Función	REGISTRO
Prioridad	Alta
Estabilidad	Alta
Descripción	Registra un tutor en la aplicación
Entrada	Nombre, email, contraseña, pregunta de seguridad y respuesta a la pregunta de seguridad.
Salida	-
Origen	Operador del sistema tutor
Destino	Sistema
Precondición	-
Postcondición	Acceso al sistema AS
Flujo de evento	**1.** Se inicia la aplicación **2.** La pantalla cambia mostrando los campos a rellenar: nombre, email, contraseña, pregunta de seguridad, respuesta a la pregunta de seguridad. **3.** Se rellenan los campos y se pulsa *Aceptar.* **4.** Acceso al sistema.
Efectos laterales	-

- Editar perfil

Función	EDITAR PERFIL
Prioridad	Alta
Estabilidad	Media
Descripción	Edita la información del tutor
Entrada	-
Salida	Mensaje
Origen	Operador del sistema tutor
Destino	Sistema
Precondición	-
Postcondición	Editar la información del tutor
Flujo de evento	**1.** Se selecciona la opción “Mi perfil”en el menú lateral. **2.** La pantalla cambia mostrando los campos editables: correo ,pregunta de seguridad y su respuesta. **3.** El tutor puede elegir: **a.** *Aceptar*: la aplicación actualiza la información en la BBDD y muestra un mensaje notificando el éxito de la operación. **b.** *Cancelar:* la aplicación no modifica nada y vuelve a la pantalla principal. **c.** *Cambiar Contraseña* Ver caso de uso: Cambiar contraseña
Efectos laterales	-

- Cambiar contraseña

Función	CAMBIAR CONTRASEÑA
Prioridad	Alta
Estabilidad	Alta
Descripción	Cambia la clave de acceso del tutor
Entrada	Nueva contraseña, repetición de la nueva contraseña
Salida	-
Origen	Operador del sistema tutor
Destino	Sistema
Precondición	-
Postcondición	Clave de acceso modificada
Flujo de evento	1. En la pestaña de Mi perfil se pulsa sobre "Cambiar contraseña" 2. La pantalla cambia mostrando dos campos para introducir la nueva contraseña y la repetición de la contraseña. 3. El tutor puede elegir: a. *Aceptar*: la aplicación actualiza la información en la BBDD. b. *Cancelar:* la aplicación no modifica nada y vuelve a la pestaña Mi perfil.
Efectos laterales	-

- Mostrar ayuda

Función	MOSTRAR AYUDA
Prioridad	Media
Estabilidad	Alta
Descripción	Muestra una guia de usuario de la *aplicación tutor*
Entrada	-
Salida	Imagen descriptiva de una determinada parte de la aplicación
Origen	Operador del sistema tutor
Destino	Sistema
Precondición	-
Postcondición	-
Flujo de evento	**1.** Se selecciona la opción "Ayuda" en el menú lateral. **2.** La aplicación muestra una lista que contiene las preguntas más frecuentes. **3.** Si se selecciona una pregunta del listado se muestra su imagen que contiene una descripción de la ayuda seleccionada.
Efectos laterales	-

- Acceso

Función	ACCESO
Prioridad	Baja
Estabilidad	Media
Descripción	Acceso a la *aplicación tutor*
Entrada	Contraseña de acceso
Salida	-
Origen	Operador del sistema tutor
Destino	Sistema
Precondición	-
Postcondición	Acceso a la *aplicación tutor*
Flujo de evento	**1.** Se inicia la aplicación **2.** Se rellena el campo contraseña **3.** El tutor puede elegir: **a.** Insertar la contraseña i. Si es correcta se accede al sistema ii. Si no es correcta se muestra el mensaje "Clave incorrecta". **b.** Pulsar en la opción "Olvidé mi contraseña": Ver caso de uso: "Olvidar contraseña"
Efectos laterales	-

- Olvidar contraseña

Función	OLVIDAR CONTRASEÑA
Prioridad	Alta
Estabilidad	Media
Descripción	Recupera el acceso a la *aplicación tutor*
Entrada	Respuesta de la pregunta de seguridad
Salida	-
Origen	Operador del sistema tutor
Destino	Sistema
Precondición	-
Postcondición	Acceso a la *aplicación tutor*
Flujo de evento	1. Se inicia la aplicación. 2. El tutor pulsa la opción *"Olvidé mi contraseña".* 3. Rellena el campo respuesta a la pregunta de seguridad. a. Si es correcta se accede al sistema b. Si no es correcta se muestra un mensaje de error "Respuesta incorrecta".
Efectos laterales	-

9.2.2 Casos de uso relacionados con los usuarios

- Crear usuario

Función	CREAR USUARIO
Prioridad	Alta
Estabilidad	Alta
Descripción	Crea un nuevo usuario
Entrada	Nombre, email, foto, centro académico, curso, DNI, tipo de perfil, notas, nombres de los padres, correos de los padres y teléfonos de los padres del usuario
Salida	Mensaje
Origen	Operador del sistema tutor
Destino	Sistema
Precondición	-
Postcondición	Se añade un usuario al sistema
Flujo de evento	**1.** En la pestaña de "Usuarios" el tutor pulsa sobre el botón de creación. **2.** La interfaz cambia mostrando los campos a rellenar, sólo son obligatorios los campos del nombre del usuario y su perfil. **3.** El tutor puede elegir: **a.** *Cancelar:* la aplicación no añade al usuario y vuelve a la pestaña de "Usuarios". **b.** *Crear*: la aplicación añade al usuario en base de datos.
Efectos laterales	-

- Editar usuario

Función	EDITAR USUARIO
Prioridad	Alta
Estabilidad	Alta
Descripción	Modifica la información sobre un usuario.
Entrada	ID y datos del usuario a modificar
Salida	Mensaje
Origen	Operador del sistema tutor
Destino	Sistema
Precondición	Usuario ya existente en el sistema.
Postcondición	Actualización de la información sobre el usuario en el sistema.
Flujo de evento	**1.** En la pestaña "Usuarios" se muestra un listado de usuarios. **2.** El tutor selecciona el usuario que quiere editar. **3.** La pantalla pasa a ser semejante a cuando se crea un nuevo usuario pero con los campos completados y con un menú en la parte superior derecha. **4.** Al pulsar sobre el campo deseado se puede cambiar el contenido. **5.** Si el tutor desea guardar los cambios debe pulsar el botón flotante de guardar, en el extremo inferior derecho de la pantalla. En caso contrario, con cambiar de pantalla sera suficiente para que se descarten las modificaciones.
Efectos laterales	-

- Eliminar usuario

Función	ELIMINAR USUARIO
Prioridad	Alta
Estabilidad	Alta
Descripción	Elimina un usuario del sistema
Entrada	ID usuario
Salida	Mensaje
Origen	Operador del sistema tutor
Destino	Sistema
Precondición	Usuario existente
Postcondición	Elimina el usuario del sistema
Flujo de evento	**1.** En la pestaña "Usuarios" se muestra un listado de usuarios. **2.** El tutor selecciona el usuario que quiere eliminar. **3.** El tutor pulsa sobre la opción de eliminar en su menú. **4.** Aparece un mensaje de confirmación. El tutor puede elegir: **a.** *No*: el mensaje se cierra y se vuelve a la pantalla principal del usuario. **b.** *Si*: el usuario es eliminado y se muestra un mensaje de éxito.
Efectos laterales	-

- Consultar usuario

Función	CONSULTAR USUARIO
Prioridad	Alta
Estabilidad	Alta
Descripción	Muestra la información de un usuario.
Entrada	ID usuario
Salida	-
Origen	Operador del sistema tutor
Destino	Sistema
Precondición	Usuario existente
Postcondición	-
Flujo de evento	**1.** La aplicación cambia de interfaz mostrando los datos personales y la puntuación del usuario, además de dos opciones, una es para la sincronización (ver caso de uso: Sincronización) y otra es un menú con una serie de posibles acciones: **a.** *Tareas*: la aplicación muestra el conjunto de tareas asignadas al usuario. (Ver caso de uso: Ver tareas usuario) **b.** *Reto*: se muestra el reto que tiene ese usuario. (Ver caso de uso: Ver reto usuario) **c.** *Eventos*: muestra los eventos a los que está invitado ese usuario. (Ver caso de uso: Ver eventos usuario) **d.** *Enviar informe*: la aplicación muestra una ventana para enviar el informe de usuario por correo. (Ver caso de uso: Enviar informe de un usuario) **e.** *Eliminar*: posibilidad de eliminar el actual usuario. (Ver caso de uso: Eliminar usuario)
Efectos laterales	-

- Consultar todos los usuarios

Función	CONSULTAR TODOS LOS USUARIOS
Prioridad	Alta
Estabilidad	Alta
Descripción	Muestra todos los usuarios del sistema
Entrada	-
Salida	-
Origen	Sistema
Destino	Sistema
Precondición	-
Postcondición	-
Flujo de evento	**1.** En la pestaña “Usuarios” se muestra un listado con el nombre y la foto de cada usuario.
Efectos laterales	-

- Enviar informe de un usuario

Función	ENVIAR INFORME DE UN USUARIO
Prioridad	Alta
Estabilidad	Alta
Descripción	Manda por un servicio de mensajería un PDF con la última información sobre el usuario seleccionado.
Entrada	ID usuario
Salida	Email con un documento PDF informativo.
Origen	Operador del sistema tutor.
Destino	Sistema
Precondición	Usuario existente.
Postcondición	-
Flujo de evento	**1.** En la pestaña "Usuarios" se muestra un listado de usuarios. **2.** El tutor se selecciona el usuario cuyo informe quiere enviar. **3.** El tutor pulsa sobre la opción de enviar informe en su menú. **4.** La aplicación genera un documento con la información personal, los últimos datos disponibles de las tareas y el reto del usuario. **5.** Se abre una ventana para enviar un correo, con el informe adjunto, para una dirección de correo proporcionada por el tutor. **6.** Se envía el correo y se vuelve a la pantalla principal de la aplicación.
Efectos laterales	-

- Sincronización

Función	SINCRONIZACIÓN
Prioridad	Alta
Estabilidad	Alta
Descripción	Establece una conexión entre la *aplicación tutor* y la *aplicación usuario* para el intercambio de información.
Entrada	ID usuario
Salida	Mensaje
Origen	Operador del sistema tutor.
Destino	*Aplicación usuario*
Precondición	Aplicaciones en la misma red WiFi
Postcondición	Recibo y envío de datos correcto.
Flujo de evento	**1.** En la pestaña "Usuarios" se muestra un listado de usuarios. **2.** El tutor selecciona el usuario con el que se quiere sincronizar. **3.** El tutor pulsa sobre la opción de sincronizar. **4.** Espera que la *aplicación usuario* correspondiente al usuario seleccionado envíe una petición de sincronización. **5.** Se establece conexión, primero procesa el mensaje recibido para después actualizar su base de datos y generar el mensaje de respuesta. **6.** Envía un mensaje con toda la información acerca de los sucesos del usuario con el que ha establecido conexión **7.** La comunicación finaliza.
Efectos laterales	Corrupción en la base de datos

9.2.3 Casos de uso relacionados con los sucesos

- Tarea
    - Ver tareas

Función	VER TAREAS
Prioridad	Alta
Estabilidad	Alta
Descripción	Muestra todas las tareas de un usuario.
Entrada	ID usuario
Salida	-
Origen	Operador del sistema tutor.
Destino	Sistema
Precondición	Usuario existente.
Postcondición	-
Flujo de evento	**1.** El tutor selecciona a un usuario del sistema. **2.** En el menú de la barra de navegación selecciona el item “Tareas”. **3.** Se abre una nueva pantalla en la que aparece un listado con todas las tareas que tiene ese usuario con sus campos más importantes.
Efectos laterales	-

- Crear tarea

Función	CREAR TAREA
Prioridad	Alta
Estabilidad	Alta
Descripción	Añade una tarea nueva al sistema
Entrada	Texto de la pregunta, texto de la alarma, hora pregunta, hora alarma, fecha, frecuencia y número de aciertos necesarios para disminuir la frecuencia de recordatorios automáticamente.
Salida	-
Origen	Operador sistema tutor
Destino	Sistema
Precondición	Usuario existente en el sistema
Postcondición	-
Flujo de evento	**1.** Desde la pantalla a la que se accede como se explica en el caso de uso "Ver tareas" se debe seleccionar el botón flotante de añadir que se encuentra en la esquina inferior izquierda. **2.** Se abre una pantalla con un formulario en el cual se rellenan los campos necesarios para crear una nueva tarea. **3.** Al final de la pantalla hay dos opciones: a. ACEPTAR: que guarda la nueva tarea en el sistema. b. CANCELAR: que descarta los cambios. **4.** Se vuelve a la pantalla "Ver tareas".
Efectos laterales	-

- Editar tarea

Función	EDITAR TAREA
Prioridad	Alta
Estabilidad	Alta
Descripción	Modifica los datos de una tarea.
Entrada	ID y datos de la tarea a modificar.
Salida	-
Origen	Operador sistema tutor
Destino	Sistema
Precondición	Tarea existente en el sistema
Postcondición	-
Flujo de evento	**1.** Desde la pantalla a la que se accede como se explica en el caso de uso "Ver tareas" se selecciona la tarea que se desee editar. **2.** Aparece una ventana de diálogo ofreciendo distintas opciones. Se debe elegir "Editar". **3.** La pantalla pasa a ser semejante a cuando se crea una nueva tarea pero con los campos completados y sin permitir la edición del campo *texto de alarma*.. **4.** Al pulsar sobre el campo deseado se puede cambiar el contenido. **5.** Al final de la pantalla hay dos opciones: a. CANCELAR: que descarta los cambios. No se modifica nada. b. ACEPTAR: que confirma los cambios. Se actualiza la información en el sistema y se muestra un mensaje de éxito.
Efectos laterales	-

- Habilitar/deshabilitar tarea

Función	HABILITAR/DESHABILITAR TAREA
Prioridad	Alta
Estabilidad	Alta
Descripción	Habilita/deshabilita una tarea de un usuario.
Entrada	ID de la tarea
Salida	-
Origen	Operador sistema tutor
Destino	Sistema
Precondición	Tarea existente
Postcondición	-
Flujo de evento	**1.** Desde la pantalla a la que se accede como se explica en el caso de uso "Ver tareas" se selecciona la tarea que se desee editar. **2.** Aparece una ventana de diálogo ofreciendo distintas opciones. Se debe elegir "Habilitar/Deshabilitar". **3.** En el listado de tareas se observará que el estado de la tarea ha cambiado porque se selecciona (o deselecciona) la checkbox que indica si esa tarea está o no habilitada para el usuario.
Efectos laterales	-

- Eliminar tarea

Función	ELIMINAR TAREA
Prioridad	Alta
Estabilidad	Alta
Descripción	Elimina una tarea (permanente o puntual) o un reto de la base de datos del tutor.
Entrada	ID tarea
Salida	Mensaje
Origen	Operador del sistema tutor.
Destino	Sistema.
Precondición	Tarea existente
Postcondición	-
Flujo de evento	**1.** Desde la pantalla a la que se accede como se explica en el caso de uso "Ver tareas" se selecciona la tarea que se desee editar. **2.** Aparece una ventana de diálogo ofreciendo distintas opciones. Se debe elegir "Eliminar". **3.** Aparece un nuevo cuadro de diálogo en el que el tutor debe confirmar que realmente quiere eliminar esa tarea. Se muestran dos opciones: a. ACEPTAR: se procede a la eliminación de la tarea. b. CANCELAR: se descarta cualquier cambio, no se hace nada. **4.** Se vuelve a la pantalla de "Ver tareas" donde, en caso de haber aceptado, la tarea ya no aparecerá en el listado.
Efectos laterales	-

- Evento
    - Crear evento

Función	CREAR EVENTO
Prioridad	Alta
Estabilidad	Alta
Descripción	Añade un evento al sistema e invita usuarios al mismo.
Entrada	Nombre del evento, hora del evento, hora de la alarma, fecha y un listado de usuarios.
Salida	-
Origen	Operador sistema tutor
Destino	Sistema
Precondición	-
Postcondición	Añade un evento al sistema e invita a los usuarios al evento.
Flujo de evento	**1.** En la pestaña de “Eventos” el tutor pulsa sobre el botón de creación.. **2.** La interfaz cambia mostrando los campos a rellenar: nombre del evento, hora del evento, hora de la alarma, fecha y una lista con todos los usuarios del sistema. **3.** El tutor puede elegir: **a.** Rellena los campos, pulsa “Crear”. Se añade el evento en base de datos y se invitan a los usuarios seleccionados al evento. **b.** Pulsa “Cancelar” vuelve a la pestaña de “Eventos”.
Efectos laterales	-

- Editar evento

Función	EDITAR EVENTO
Prioridad	Alta
Estabilidad	Alta
Descripción	Modifica los datos de una evento puntual.
Entrada	ID, nombre del evento, hora del evento, hora de la alarma, fecha y un listado de usuarios.
Salida	-
Origen	Operador sistema tutor
Destino	Sistema
Precondición	Evento existente en el sistema.
Postcondición	Actualiza la información de un evento y las invitaciones de los usuarios al evento.
Flujo de evento	**1.** En la pestaña de "Eventos" el tutor pulsa sobre el botón de creación.. **2.** La interfaz cambia mostrando los campos editables con la información del evento: nombre, hora, hora de la alarma, fecha y un listado con todos los usuarios del sistema, mostrando para cada uno de ellos si está o no invitado al evento, así como su asistencia al mismo. **3.** El tutor puede elegir: a. Edita los campos, pulsa la opción "Guardar". Se actualiza el evento en base de datos y su listado de invitados. b. Pulsa la opción "Cancelar" vuelve a la pestaña de "Eventos".
Efectos laterales	-

- Consultar eventos del usuario

Función	CONSULTAR EVENTOS DEL USUARIO
Prioridad	Alta
Estabilidad	Alta
Descripción	Se muestra la información de todos los eventos de un usuario en el sistema.
Entrada	ID usuario
Salida	Datos de los eventos del usuario.
Origen	Operador sistema tutor
Destino	Sistema
Precondición	-
Postcondición	-
Flujo de evento	**1.** En la pestaña "Usuarios" el tutor selecciona un usuario del listado. **2.** Se pulsa sobre el menú del usuario en la parte superior derecha. **3.** Se muestran dos posibles opciones: a. Si el usuario no tiene eventos saldrá un mensaje en la pantalla notificando dicho suceso. b. El usuario tiene al menos un evento, por lo tanto se muestra un listado con el nombre, fecha, hora y asistencia al evento.
Efectos laterales	-

- Consultar todos los eventos

Función	CONSULTAR TODOS LOS EVENTOS
Prioridad	Alta
Estabilidad	Alta
Descripción	Se muestra la información de todos los eventos del sistema.
Entrada	-
Salida	Datos de los eventos del sistema.
Origen	Operador sistema tutor
Destino	Sistema
Precondición	-
Postcondición	-
Flujo de evento	**1.** El tutor pulsa sobre la pestaña de “Eventos”. **2.** Se muestra un listado con el nombre, fecha y hora del evento.
Efectos laterales	-

- Eliminar evento

Función	ELIMINAR EVENTO
Prioridad	Alta
Estabilidad	Alta
Descripción	Elimina un evento del sistema
Entrada	ID del evento
Salida	-
Origen	Operador del sistema tutor.
Destino	Sistema.
Precondición	Evento existente.
Postcondición	Se elimina un evento de la base de datos
Flujo de evento	**1.** En la pestaña “Eventos” se muestra un listado de eventos. **2.** Se selecciona un evento. **3.** Se pulsa sobre la opción de eliminar. **4.** Aparece un mensaje de confirmación. El tutor tiene dos opciones: a. Confirmar: el evento se elimina de base de datos. Se vuelve a la pestaña de “Eventos”. b. Cancelar: se cierra el mensaje de confirmación. Se vuelve al detalle del evento.
Efectos laterales	-

- Reto
    - Crear reto

Función	CREAR RETO
Prioridad	Media
Estabilidad	Alta
Descripción	Añade un reto asignado a usuario del sistema
Entrada	ID usuario junto al texto y premio del reto
Salida	Mensaje
Origen	Operador sistema tutor
Destino	Sistema
Precondición	Usuario existente
Postcondición	Crea un reto y lo añade a la base de datos
Flujo de evento	**1.** En la pestaña “Usuarios” se muestra un listado de usuarios. **2.** El tutor selecciona el usuario al cual quiere añadir un reto. **3.** El tutor pulsa sobre la opción reto en su menú. **4.** Aparece una pantalla donde se puede introducir el texto y un premio opcional para el nuevo reto. **5.** El tutor puede elegir: **a.** Cancelar: el reto no se crea y vuelve a la pantalla principal del usuario. **b.** Crear: el reto se almacena en el sistema y aparece un mensaje de éxito.
Efectos laterales	-

- Ver reto

Función	VER RETO
Prioridad	Alta
Estabilidad	Alta
Descripción	En función de si el usuario posee o no un reto aparece una pantalla de creación o una informativa.
Entrada	ID usuario
Salida	Datos del reto o pantalla de creación.
Origen	Operador del sistema tutor
Destino	Sistema
Precondición	Usuario existente
Postcondición	-
Flujo de evento	**1.** La aplicación consulta si el usuario tiene un reto ya asignado o no: **a.** Tiene un reto: se muestra una pantalla con el texto del reto, su posible premio y una barra con su progreso; además de dos opciones, una sirve para eliminar el reto (ver caso de uso: Eliminar reto) y la otra es un menú con las siguientes funcionalidades: **i.** Volver al usuario: regresa a la pantalla principal del usuario **ii.** Tareas: la aplicación muestra el conjunto de tareas asignadas al usuario. (Ver caso de uso: Ver tareas usuario) **iii.** Evento: muestra los eventos a los que está invitado ese usuario. (Ver caso de uso: Ver eventos usuario) **iv.** Enviar informe: la aplicación muestra una ventana para enviar el informe de usuario por correo. (Ver caso de uso: Enviar informe de un usuario) **b.** No tiene un reto: lanza una pantalla para poder añadir un reto al usuario. (Ver caso de uso: Crear reto)
Efectos laterales	-

- Eliminar reto

Función	ELIMINAR RETO
Prioridad	Media
Estabilidad	Alta
Descripción	Elimina el reto de un usuario.
Entrada	ID reto
Salida	Mensaje
Origen	Operador del sistema tutor.
Destino	Sistema.
Precondición	Reto existente
Postcondición	Se elimina un reto de la base de datos.
Flujo de evento	**1.** En la pestaña “Usuarios” se muestra un listado de usuarios. **2.** El tutor selecciona el usuario al cual quiere eliminar su reto. **3.** El tutor pulsa sobre la opción reto en su menú. **4.** El tutor pulsa la opción de eliminar. **5.** Aparece un mensaje de confirmación. El tutor puede: **a.** Cancelar: no se elimina el reto del usuario y se vuelve a la pantalla principal del reto de ese usuario. **b.** Confirmar: se destruye el reto y aparece un mensaje de éxito.
Efectos laterales	-

CAPÍTULO 10: Conclusión y trabajo futuro

"Me gustan más los sueños del futuro
que las historias del pasado."
THOMAS JEFFERSON

En este capítulo se exponen las conclusiones de nuestro proyecto y una lista de posibles implementaciones como trabajo futuro.

10.1 Conclusiones

Al finalizar este proyecto se puede concluir que se hemos cumplido el principal objetivo, es decir, desarrollar una herramienta que realmente facilita la labor de los profesionales que trabajan con personas Asperger y mejora la calidad de vida de las propias personas con este síndrome.

Además hemos completado una serie de metas que nos habíamos propuesto a nivel personal como son **la realización de un proyecto en el sistema operativo Android**, un lenguaje que queríamos aprender, ya que teníamos un gran interés por las tecnologías móviles y que no teníamos ninguna experiencia previa de él; **la comunicación entre las aplicaciones sin un servidor**, una problemática que desconocíamos por completo y que hemos solucionado mediante el uso de sockets en una red WiFi; y por último **queríamos emplear el máximo de los conocimientos que habíamos adquirido** durante estos años en el grado de Ingeniería del Software especialmente el uso de patrones software, vistos en las asignaturas *de Ingeniería del Software* y *Modelado del Software*; facilitar la creación de las bases de datos mediante el mapeo de las clases Java con frameworks, visto también en la asignatura de *Modelado del Software*, y un correcto diseño de las pantallas de las dos aplicaciones, visto en la asignatura optativa *Interfaces de Usuario*. Estamos muy orgullosos de haber puesto en práctica gran parte de las asignaturas de la carrera: *Bases de Datos, Tecnologías de la Programación, Ampliación de Bases de Datos, Aplicaciones Web,* etc.

Para diseñar las aplicaciones según las necesidades de las personas Asperger y sus tutores fuimos asesorados por expertos de la Asociación Implica. Gracias a esto, tuvimos la oportunidad de vivir la experiencia de tener un cliente real. Hemos aprendido todo lo que conlleva la interacción con el cliente: por una parte, los continuos cambios de requisitos que pueden hacer

evolucionar una idea o pueden retrasarla, y por otra parte, la motivación que supone realizar un proyecto con un fin real sabiendo que será utilizada para bien de muchas personas.

Tras finalizar la implementación se reservó una temporada en la planificación del proyecto para la realización de una serie de pruebas tanto internas como externas. A nivel interno, hemos probado la robustez del sistema ante varios casos extremos, solucionando los errores que nos encontrábamos. A nivel externo, hemos trabajado con la Asociación Implica instalando las aplicaciones en dispositivos de algunas personas Asperger para comprobar su correcto funcionamiento, así como posibles errores o mejoras a realizar. Como resultado de estas pruebas se han obtenido tres versiones de las aplicaciones. La última y definitiva es la 3.0 que está disponible en Google Play.

Queremos compartir al máximo este trabajo y dar la posibilidad de que cualquier persona pueda adaptarlo a sus necesidades para aprovechar sus ventajas, como la sencillez de la interfaz y la novedad de la idea, que pueden ser muy útiles para otro tipo de colectivos como los Alzheimer o como un método de seguimiento en las escuelas. Para ello hemos subido el proyecto a un repositorio en GitHub. Como parte de este interés por difundir nuestra idea hemos participado en la "Convocatoria Soluciones Digitales 2016" de la fundación Orange [14] que está centrada en fomentar el desarrollo de proyectos tecnológicos que ayuden a personas dentro del espectro del autismo.

A continuación, se encuentran los enlaces a los repositorios donde se puede acceder al código fuente de las dos aplicaciones:

- Enlace GitHub *aplicación usuario*: https://github.com/jeff1191/AS_USUARIO
- Enlace GitHub *aplicación tutor*: https://github.com/jeff1191/AS_TUTOR

A continuación, se encuentran los enlaces de Google Play donde ya se encuentran disponibles la *aplicación tutor* y la *aplicación usuario* para ser descargadas y utilizadas gratuitamente:

- Enlace Google Play *aplicación usuario*: https://play.google.com/store/apps/details?id=es.ucm.as_usuario&hl=es

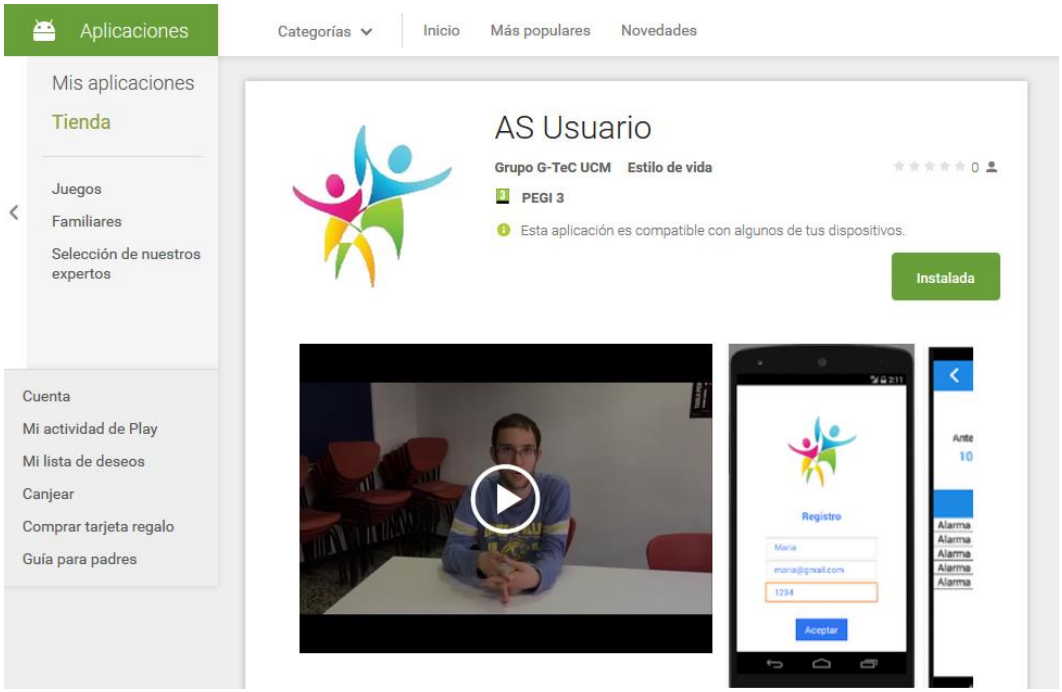


Figura 86: Imagen de la *aplicación usuario* en Google Play

- Enlace Google Play *aplicación tutor*: https://play.google.com/store/apps/details?id=es.ucm.as_tutor&hl=es

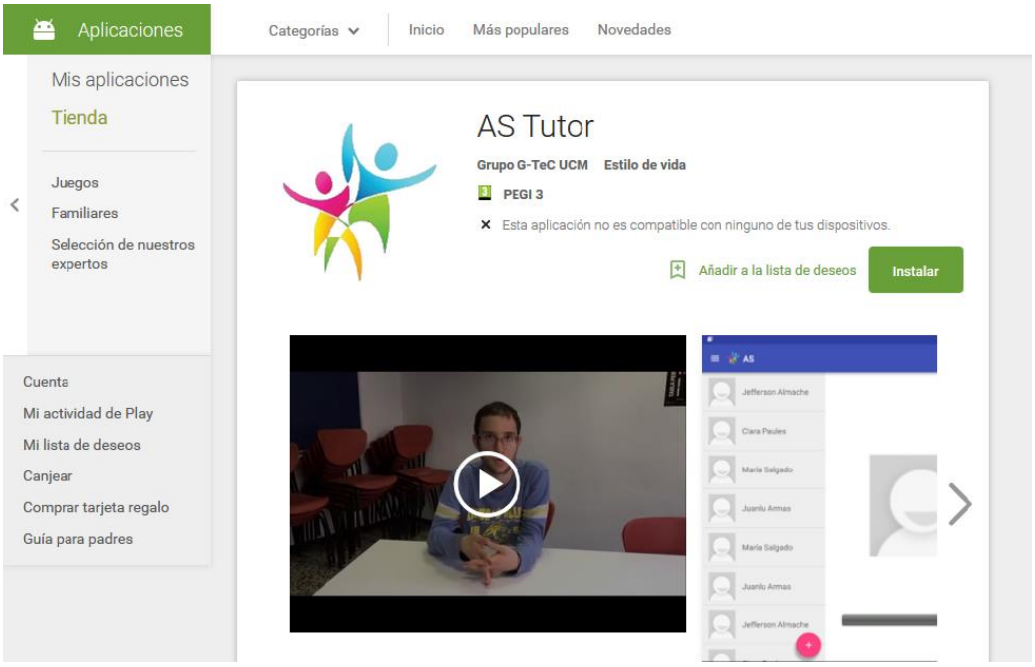


Figura 87: Imagen de la aplicación usuario en Google Play

10.2 Trabajo futuro

A continuación, se citan algunas de las posibles mejoras que se nos han ocurrido a nosotros y a los especialistas de la Asociación Implica durante el desarrollo y las pruebas de este proyecto.

Todas ellas son factibles, pudiendo incluirse cómodamente debido a la arquitectura modular y el uso de patrones con los que hemos diseñado ambas aplicaciones.

- *Posibilidad de parametrizar la frecuencia y el modo de responder a los retos*. La idea es habilitar en la *aplicación tutor* la opción de elegir el número de veces al día que un usuario puede contestar un reto y si se debe contestar mediante un texto o con los botones por defecto.

- *Habilitar la recuperación los datos*. En ambas aplicaciones disponer de una opción de generar copias de seguridad y almacenarlas en un lugar seguro, para que, en caso de pérdida o borrado accidental del dispositivo, la información del usuario y del tutor no desapareciera.

- *Transmisión de información a través de redes móviles*. Establecer la comunicación entre las aplicaciones mediante redes de telefonía móvil, en vez del actual uso de la tecnología WiFi.

- *Posibilidad de responder las tareas más tarde*. Crear una opción en el menú principal de la *aplicación usuario* para que en caso de que éste no pudiera contestar alguna pregunta en el momento, se pudiera entrar más tarde para contestar todas las tareas sin respuesta.

- *Cifrado del contenido de base de datos*. Esta medida aumentaría la seguridad en nuestras aplicaciones y contribuiría a preservar la privacidad de los usuarios.

- *Adaptación de las aplicaciones al sistema operativo iOS*. De este modo se permite el uso a todas aquellas personas que no dispongan de un dispositivo Android lo cual supondría un aumento en la popularidad de nuestro sistema.

CHAPTER 11: Conclusions and future work

In this chapter the conclusions of our project and a list of possible future implementations are presented.

10.1 Conclusions

Upon completion of this project it can be concluded that its main objective has been fulfilled, i.e. to develop a tool that really facilitates the work of professionals working with people with Asperger and improves the quality of life of the people with this syndrome.

We have also completed a number of goals that we had proposed ourselves personally such as **the realization of a project on the Android operating system**, a language that we wanted to learn since we had a great interest in mobile technologies and we had no previous experience on it; **communication between applications without a server**, a problem that we previously did not know at all and that we have solved by using sockets in a WiFi network; and finally we wanted to **use the maximum of the knowledge we had acquired** over these years studying the Software Engineering degree, especially the use of software patterns, seen in the subjects *Software Engineering* and *Software Modeling*; facilitating the creation of databases by mapping Java classes with frameworks, also seen in the course of *Software Model*ing, and a proper design of the screens of the two applications, seen in the optional subject User Interfaces. We are very proud to have put many of the subjects of the grade into practice: *Databases, Programming Technologies, Databases Expansion, Web Applications*, etc.

In order to design the applications according to the needs of people with Asperger and their tutors, we were advised by experts from the Implica Association. Thanks to this, we had the opportunity to live the experience of having a real customer. We have learned all the aspects that customer interaction involves. On the one hand, the constant changes of requirements that can make an idea evolve or can delay it, and on the other hand, the motivation that comes from a project with a real purpose, knowing that it will be used for the good of many people.

After completing the implementation, a period of time was reserved in the project planning to carry out a series of tests both internal and external. At an internal level, the robustness of the system has been tested for some extreme cases, solving the mistakes found. Externally, we worked with the Implica Association, installing the applications in the devices of some people with Asperger, to check their proper functioning as well as possible errors or any

improvements to be made. As results of these tests were obtained three versions of the applications. The last and final is the version 3.0, which is available on Google Play.

We want to share the most of this work and give the possibility to anyone to suit theirs needs and to exploit its advantages, such as the simplicity of the interface and the novelty of the idea, which can be very useful for other groups such as Alzheimer's or as a method of tracking in schools. So we have uploaded the project to a repository on GitHub. As part of this interest in spreading our idea we have participated in the "Digital Solutions Contest 2016" of the Orange's Foundation [14] that is focused on promoting the development of technological projects that help people within the autism spectrum.

Here are links to repositories where you can access the source code of the two applications:

- *User applicaction* GitHub link: https://github.com/jeff1191/AS_USUARIO
- *Tutor application* GitHub link: https://github.com/jeff1191/AS_TUTOR

Below are Google Play links where *tutor application* and *user application* are available to be used for free:

- *User applicaction* Google Play link: https://play.google.com/store/apps/details?id=es.ucm.as_usuario&hl=es

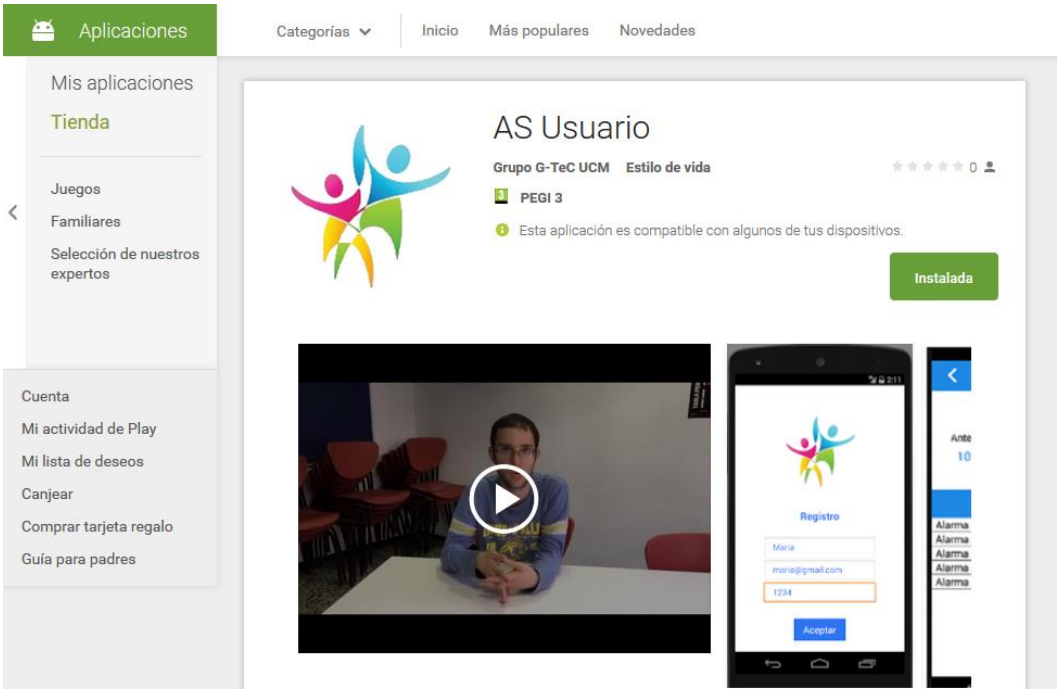


Figura 86: Image of the *user application* at Google Play

- *Tutor application* Google Play link: https://github.com/jeff1191/AS_TUTOR
https://play.google.com/store/apps/details?id=es.ucm.as_tutor&hl=es

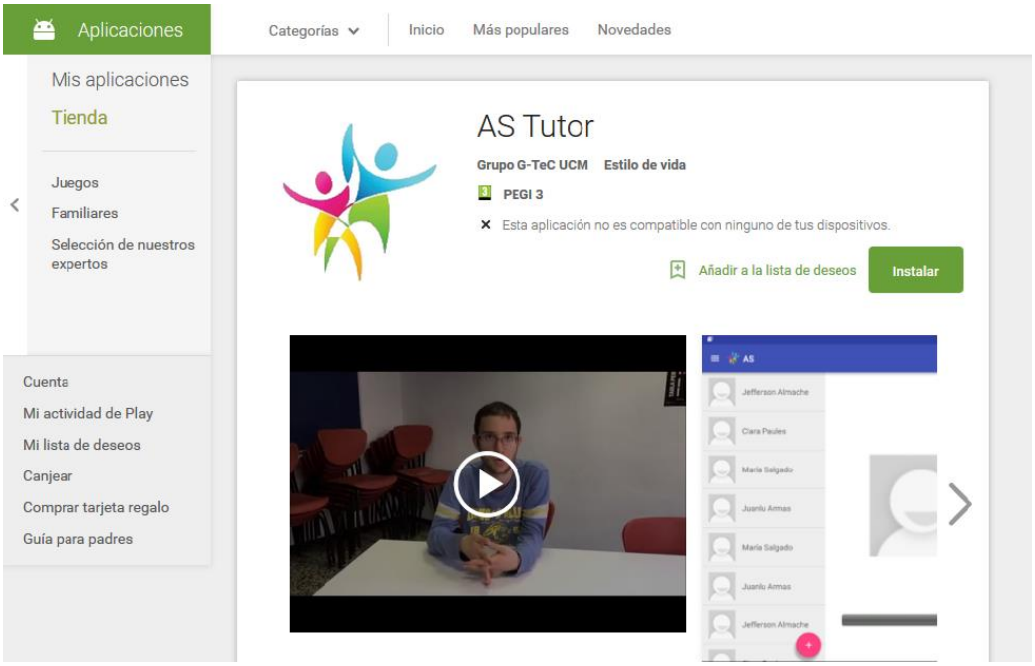


Figura 87: Image of the *tutor application* at Google Play

10.2 Future work

Below are some possible improvements thought by the specialist of the Implica Association and us during the development and testing of this project. All of them are feasible and can be included easily due to the modular architecture and the use of patterns used in this application.

- *Possibility to parameterize the frequency and the way of responding to the challenges*. The idea is to enable in the *tutor application* the option to choose the number of times per day that a user can answer a challenge and whether to answer via a text or with the buttons by default.

- *Enable data recovery*. To have an option in both applications to create backups and store them in a safe location, so that in case of loss or accidental deletion of the device, the user's and the tutor's information does not disappear.

- *Transmission of information through mobile networks*. To establish the communication between both applications via mobile phone networks, instead of the current use of the WiFi technology.

- *Possibility to answer tasks later*. To create an option on the main menu of the *user application* to allow the user to respond all the previous unanswered questions, in case he or she could not have answered them before.

- *Encryption of the database content*. This measure would increase security in both applications and would also help preserve the privacy of users.

- *Adaptation of the applications to the iOS operating system*. This way any person, even those who are not in possession of an Android device, would have access to these applications, which would mean an increase in the popularity of our system.

CAPÍTULO 12: Tareas desarrolladas

"La acción es la llave fundamental de todo éxito."

PABLO PICASSO

El equipo que ha desarrollado este proyecto está formado por tres alumnos de Grado en Ingeniería del Software. Realmente todo el proyecto se ha realizado conjuntamente: el diseño de la idea, la especificación de requisitos, el diseño de las bases de datos, la implementación, las pruebas y la redacción de la memoria. A continuación, se detallan algunas de las tareas concretas que ha desempeñado cada miembro y se destacan las aportaciones más importantes de cada uno. A todo lo que aquí se expone hay que sumar intensas horas de aprendizaje de tecnologías nuevas para nosotros, especialmente el desarrollo Android y el intercambio de información a través de sockets.

12.1 Jefferson Almache Montoya

Para desarrollar el sistema AS, Jefferson se incorporó a la idea e iniciativa de María de realizar un proyecto solidario relacionado con las personas autistas. Una vez que se dio forma a la idea con ayuda de Dra. Guadalupe Miñana Ropero, Dra. Victoria López López y la asociación Implica, Jefferson empezó a estudiar Android el verano del 2015, lenguaje escogido para implementar la *aplicación tutor* y la *aplicación usuario*. Después, cuando empezó el periodo académico 2015/2016 y teniendo claro los objetivos, marco tecnológico y ámbito del proyecto, Jefferson junto con sus compañeros colaboró con la especificación de requisitos, planificación y esquemas iniciales sobre el aspecto del sistema AS.

Jefferson propuso en la fase de diseño implementar la arquitectura multicapa para el proyecto utilizando los conocimientos obtenidos a lo largo de la carrera. Diseñó en primera instancia la arquitectura del sistema AS utilizando patrones de ingeniería de software, para ello emplea la herramienta *IBM Rational Software Architect 8.0.* Además propone usar un Object-relational mapping (ORM) para optimizar y facilitar la gestión de la base de datos, primeramente se trata de usar Java Persistence API(JPA) pero al no ser compatible con Android se descartó la idea utilizando ORMLite finalmente.

En la fase de implementación se decide comenzar por la *aplicación usuario* donde Jefferson desarrolló el launch screen, gestión de eventos y la configuración de la aplicación en el cual se destaca su trabajo en la personalización del color. Posteriormente, para implementar la *aplicación tutor* desarrolla en un principio la interfaz gráfica donde se sigue el diseño de material design de Google para ello se utiliza fragments con el apoyo de un drawerLayout para que de un aspecto atractivo, además él se encarga de la gestión de eventos donde cobra especial dificultad el manejo de la relación N M que existe entre eventos y usuarios, también se encarga del cifrado de la clave de acceso de la *aplicación tutor,* el launch screen y las pantallas de ayuda. Una vez se termina la fase de implementación de la aplicación tutor y usuario se reúne él y sus compañeros con la asociación Implica donde se cambiarán algunos requisitos que posteriormente se desarrollaran en un periodo breve.

La sincronización entre la *aplicación usuario* y la *aplicación tutor* cobra una relevancia importante en nuestro sistema, tanto Jefferson como Juan Luis y María colaboran conjuntamente para solventar esta fase de desarrollo, donde tienen que reunirse continuamente por dos motivos: para sincronizar las dos aplicaciones necesitan estar en la misma red Wifi, solo tienen una tablet facilitada por la asociación Implica.

Después de terminar con la fase de implementación del sistema AS, Jefferson realizó una revisión del código, dándose cuenta de que no se respetaba la arquitectura multicapa, por lo que decidió hacer una reestructuración del código. Para ello, corrigió junto con Juan Luis y María los errores que se descubrieron. Una vez que terminaron esta labor, realizaron pruebas funcionales e implantaron las aplicaciones en la Asociación Implica.

Para el desarrollo del sistema AS Jefferson ha destacado por mantener la modularidad en la implementación respetando el modelo diseñado. Sus conocimientos en Android han sido esenciales para llevar a cabo este proyecto. Además ha aportado aspectos importantes en el desarrollo de software debido a sus conocimientos técnicos y a su experiencia en dirección de proyectos a lo largo de sus estudios. También se destaca su puntualidad e implicación en las reuniones que se han llevado a cabo con la Asociación Implica y con la Dra. Guadalupe Miñana Ropero, tutora de este proyecto.

En la redacción de la memoria ha desarrollado el capítulo 5 donde se describe arquitectura del sistema AS y la descripción de los patrones de ingeniería de software que se han utilizado, asimismo ha participado en el capítulo 9 donde se detallan los casos de uso de la *aplicación tutor* y la *aplicación usuario*, finalmente realiza el anexo B de este documento donde se explica

la planificación que hemos seguido durante el desarrollo del proyecto. Tanto Jefferson como sus compañeros participaron en el curso "Taller de gestores bibliográficos, fuentes de información y citas" que se impartió a principio de curso y que ha sido muy práctico y útil para la redacción de esta memoria. También ha participado en la revisión de este documento y ha aportado ideas en los distintos capítulos de la misma.

12.2 Juan Luis Armas Perona

Los tres integrantes de este TFG eligieron desarrollar la idea de María de realizar un proyecto a través del cual ayudar a personas discapacitadas. Mediante la Dra. Guadalupe Miñana Ropero, la Dra. Victoria López López y la Asociación Implica de Madrid dieron una forma a esa idea antes del fin de curso 2014/2015. Puesto que acordaron que el proyecto iba a ser desarrollado en una tecnología móvil y el sistema operativo predominante en el mercado es Android, decidieron que los tres integrantes del grupo dedicaran el verano a investigar este lenguaje, como preparación previa a la implementación del proyecto, puesto que ninguno tenía una experiencia previa con él. De esta manera Juan Luis desarrolló varias aplicaciones sencillas a modo de prácticas guiándose en tutoriales online para tener un mejor conocimiento de los elementos y el desarrollo en Android.

Durante los primeros meses del curso 2015/2016, estuvieron diseñando la idea y realizando bocetos de las posibles pantallas de las aplicaciones, como equipo se repartieron las tareas a realizar de forma equitativa. Tanto Juan Luis, como María y Jefferson estuvieron buscando información acerca de cómo podían implementar los aspectos de este proyecto: la comunicación entre aplicaciones, bases de datos relacionales en Android, formas de generar una aplicación, etc. Así mismo también empezaron a redactar esta memoria final. Juan Luis se encargó de redactar el capítulo cuarto, referente a las tecnologías utilizadas, así como una serie de diagramas de actividad, que posteriormente se desecharon por estar obsoletos, y parte de los casos de uso de la idea inicial.

Una vez iniciada la fase de implementación Juan Luis se encargó en la *aplicación usuario* de varios aspectos como es el diseño inicial de la pantalla de personalización del usuario, así como su funcionalidad; la creación de la pantalla de registro; la gestión del reto, que incluye la creación de la pantalla asociada a este así como su funcionalidad; la colaboración junto Jeff y María para conseguir una comunicación con la *aplicación tutor*; la realización de todas las pantallas de ayuda; y el sistema de notificaciones, en colaboración con María, que ha supuesto

todo un desafío por la serie de problemas que ha dado pero finalmente realiza los recordatorios, las preguntas y gestiona las respuestas de los sucesos correctamente.

La participación de Juan Luis en la *aplicación tutor* ha consistido en la implementación de una parte de la gestión de los usuarios, como es su creación, su modificación, su eliminación, así como la gestión de sus posibles retos. Además, también ha participado con Jefferson y María en la realización de la sincronización entre ambas aplicaciones. Al igual que con la anterior aplicación primero se desarrollaron las pantallas, divididas equitativamente entre el grupo, para después realizar su funcionalidad.

En las últimas semanas de la implementación de las aplicaciones terminaron la memoria: reescribiendo los capítulos que ya habían sido escritos debido a los cambios en las aplicaciones (tarea en la Juan Luis participó revisando los casos de uso que tenía asignados así como las tecnologías usadas) y escribiendo los que faltaban como el capítulo sexto referente a la *aplicación usuario*, el cual redactó Juan Luis.

A lo largo del proyecto Juan Luis se ha encargado de realizar pruebas internas en sendas aplicaciones para comprobar su correcto funcionamiento, con especial interés y énfasis en todo lo relevante a las notificaciones, no solo por ser uno de los encargados de implementarlas sino porque son una parte vital de nuestro proyecto de cuyo funcionamiento depende el éxito de la monitorización del sistema AS, como resultado de hacer estas pruebas se han detectado fallos que han sido comunicados a los otros miembros del equipo para que fueran subsanados entre todos lo antes posible.

También se han realizado una serie de pruebas externas, Jefferson junto con Juan Luis, han ido a la Asociación Implica para probar en situaciones reales ambas aplicaciones, instalando la *aplicación usuario* en dispositivos de varios alumnos y analizando las respuestas recibidas en la *aplicación tutor* para averiguar la fiabilidad del sistema AS, descubriendo fallos y escuchando posibles mejoras, todos solucionados y todas añadidas antes de la versión final.

La responsabilidad de escribir el capítulo décimo que trata sobre las conclusiones de su trabajo, así como las posibles tareas a realizar para mejorar este proyecto recayó sobre Juan Luis, sin embargo, tanto Jefferson como María ofrecieron sugerencias que se incorporaron al resultado final.

Una de las ideas con las que empezaron este trabajo fue el de intentar que su proyecto ayudará a los demás, por esta razón se inscribieron en una convocatoria de la fundación Orange dedicada a promover proyecto que ayuden a personas dentro del espectro del autismo; gran parte del esfuerzo para rellenar la solicitud fue realizado por María, sin embargo, Juan Luis también colaboró con la realización del presupuesto de la aplicación y rellenando algunos campos de la solicitud.

Juan Luis también ha colaborado junto a María y Jefferson en la revisión final de este documento, en la adaptación de la autorización de difusión que aparece en las primeras páginas y en la realización de los diagramas de entidad relación que aparecen al final de los capítulos de las aplicaciones.

12.3 María Salgado Iturrino

María fue la impulsora de la idea inicial de realizar un proyecto de carácter solidario con el que ayudar a personas discapacitadas, aunque no tenía pensado nada concreto. Para desarrollar un TFG de estas características decidió formar equipo con sus dos compañeros Juan Luis y Jefferson con los que había trabajado desde el segundo curso del grado con excelentes resultados. Tras contar con el apoyo de sus compañeros fueron juntos a ver a la Dra. Guadalupe Miñana Ropero a la que tenían en gran estima desde que cursaron una asignatura que ella impartía. La directora del proyecto, en colaboración con la Dra. Victoria López López, encontró a la Asociación Implica gracias a la cual se concretó la idea de hacer el sistema AS, formado por dos aplicaciones Android.

Durante el verano los tres alumnos se dedicaron a investigar la factibilidad de las ideas que se les habían ocurrido para las aplicaciones. María se documentó sobre el Asperger y comenzó a entrenarse en el desarrollo Android a través de cursos gratuitos online.

El primer diseño oficial de la interfaz fue realizado con la herramienta Pencil por María y Juan Luis. Los alumnos prepararon una presentación de Power Point explicando todas las pantallas y después la expusieron a los expertos de la Asociación Implica. María realizó las actas de las primeras reuniones con la asociación y se encargó durante todo el proyecto de solicitar al cliente lo que iban necesitando: las baterías de tareas asociadas a los perfiles determinados de los usuarios, dudas, tonos de notificaciones, frecuencias de los recordatorios, etc.

Periódicamente los alumnos se reunían con la tutora para mostrarle el trabajo realizado y plantear las dificultades que iban surgiendo. María insistía en la realización puntual de estas reuniones y en el cumplimento de los objetivos propuestos para cada plazo. Se puede decir que ella ha supuesto una valiosa aportación a la eficiencia y puntualidad del desarrollo del proyecto debido a su buena capacidad de organización y a su alto sentido de la responsabilidad.

Tras las fases de diseño, planificación y documentación comenzó la fase de implementación. A continuación, se detalla el trabajo específico de María en ambas aplicaciones. Estas tareas se realizaron siempre en colaboración con Juan Luis y Jefferson.

En la *aplicación usuario* implementó: todo lo referente a gestión de tareas y puntuación que se mostraba en la pantalla "¿Cómo vas?" y en la pantallas principal; la generación de informes en PDF; el envío del informe por correo; la funcionalidad para acceder a las pantallas de ayuda; la personalización del tono de las notificaciones; el arranque de la aplicación en base a si era la primera vez que se ejecutaba en el dispositivo (pantalla de registro) o no (pantalla principal); la sincronización con la aplicación tutor en colaboración con Juan Luis y Jefferson; el sistema automático de cambio de frecuencia en base a las respuestas de cada tarea; y también colaboró con Juan Luis en parte de la implementación del sistema de notificaciones diseñando el autoarranque del sistema de notificaciones al encender el dispositivo y arreglando algunos errores que mejoraron el funcionamiento como actualizar las notificaciones justo después de una sincronización en vez de esperar hasta el día siguiente.

Se destaca el trabajo de María en el diseño de los layouts de las pantallas de la interfaz utilizando algunas características XML que las hacen adaptables a dispositivos de cualquier tamaño.

En la *aplicación tutor* implementó: la gestión de tareas de los usuarios con las distintas pantallas para crear, editar o mostrar todas las tareas de un usuario; el acceso con clave a la aplicación; la pantalla de la configuración del perfil del profesor; la funcionalidad de ayuda también ayudó a Jefferson y colaboró con Juan Luis con la sincronización con la *aplicación usuario*; y realizó la mitad de las explicaciones de las distintas preguntas a las que se responde en el apartado de ayuda. María investigó sobre *Material Design* y aportó estos conocimientos para, junto a sus compañeros, diseñar la interfaz de la *aplicación tutor* utilizando elementos de este estilo.

Tras finalizar la implementación, Jefferson señaló algunos errores que habíamos cometido por no respetar totalmente la modularidad de la arquitectura multicapa elegida. María se encargó

de corregirlos en la *aplicación usuario* realizando una separación absoluta entre el modelo y la vista.

A lo largo de todo el trabajo María ha redactado, además del presente capítulo, los siguientes capítulos de este documento: el resumen, la introducción, el estado del arte, el capítulo que detalla la *aplicación tutor*, el capítulo que explica la comunicación entre los dos tipos de aplicación, algunos casos de uso y el anexo A, con todos los diagramas de clase que se muestran en el anexo. Además, ha realizado todas las traducciones al inglés requeridas por la normativa: abstract, introduction and conclusions and future work. Ella se ha encargado de dar el formato correcto a este documento adaptándolo a la normativa basándose en memorias de otros años disponibles en la Biblioteca de la Facultad y en otros recursos online. Para la correcta redacción de esta memoria los tres alumnos participaron en el curso "Taller de gestores bibliográficos, fuentes de información y citas" que se impartió a principio de curso, lo cual realmente ha sido muy útil. Dado que este documento ha sido redactado de manera colaborativa utilizando Google Drive.

Otro papel desempeñado por esta alumna ha sido la subida y el mantenimiento de las aplicaciones a Google Play para posibilitar el desarrollo de pruebas con usuarios Asperger reales llevadas a cabo por los otros integrantes del equipo.

Como muestra de la gran involucración personal de los alumnos en este proyecto cabe destacar su participación en la "Convocatoria Soluciones Digitales 2016" de la fundación Orange [14]. María se informó de todo lo necesario mediante llamadas telefónicas y realizó los documentos necesarios para participar además de inscribir el proyecto y encargarse de que la Asociación Implica completara las secciones del formulario que les correspondía para poder participar. De cara también a este concurso los tres alumnos grabaron algunas tomas y María editó un video promocional del sistema AS que ahora se encuentra disponible en YouTube y que ha sido también utilizado en Google Play (https://www.youtube.com/watch?v=VyQzU5faB7I).

Figura 90: Los tres alumnos autores de este proyecto.

Bibliografía y referencias

[1] FUNDACIÓN ORANGE. Aplicaciones Móviles Para Personas Con Autismo. Disponible en: http://ohmyphone.orange.es/iphone/aplicaciones/aplicaciones-moviles-para-personas-con-autismo.html

[2] EL PAÍS. Las 50 Mejores Apps De Salud En Español. Disponible en: http://tecnologia.elpais.com/tecnologia/2014/03/19/actualidad/1395249887_701638.html

[3] MOBILE WORLD CAPITAL BARCELONA. Dos Apps Para Enseñar Jugando a Niños Con Autismo Y Asperger. Disponible en: http://www.huffingtonpost.es/2015/03/06/apps-ninos-autismo-asperger_n_6809036.html

[4] Anonymous Apple Crece En Europa, Pero Android Sigue Liderando El Mercado. Disponible en: http://www.reasonwhy.es/actualidad/sociedad-y-consumo/android-sigue-liderando-el-mercado-en-espana-aunque-apple-recorta

[5] Anonymous Android Platform Version Distribution Numbers .2015. Disponible en: http://www.androidpolice.com/2015/08/03/android-platform-version-distribution-numbers-for-june-and-july-now-up-lollipop-up-to-18-1-of-all-devices/

[6] ANDROID DEVELOPERS. Dashboards . Disponible en: http://developer.android.com/intl/es/about/dashboards/index.html

[7] GOOGLE. Material Design. Disponible en: https://www.google.com/design/spec/material-design/introduction.html

[8] ANDROID DEVELOPERS. API De Android 5.0. Disponible en: http://developer.android.com/intl/es/about/versions/android-5.0.html

[9] ANDROID DEVELOPERS. Fragments. Disponible en: http://developer.android.com/intl/es/guide/components/fragments.html

[10] ANDROID DEVELOPERS. Activity. Disponible en: http://developer.android.com/intl/es/reference/android/app/Activity.html

[11] Androideity. Sockets en android. https://github.com/Androideity/Sockets-en-Android-Server/blob/HEAD/SimpleServer-Socket/src/test/Sockettest/Server.java

[12] Anonymous Socket De Internet.2016. Disponible en: https://es.wikipedia.org/w/index.php?title=Socket_de_Internet&oldid=90151707

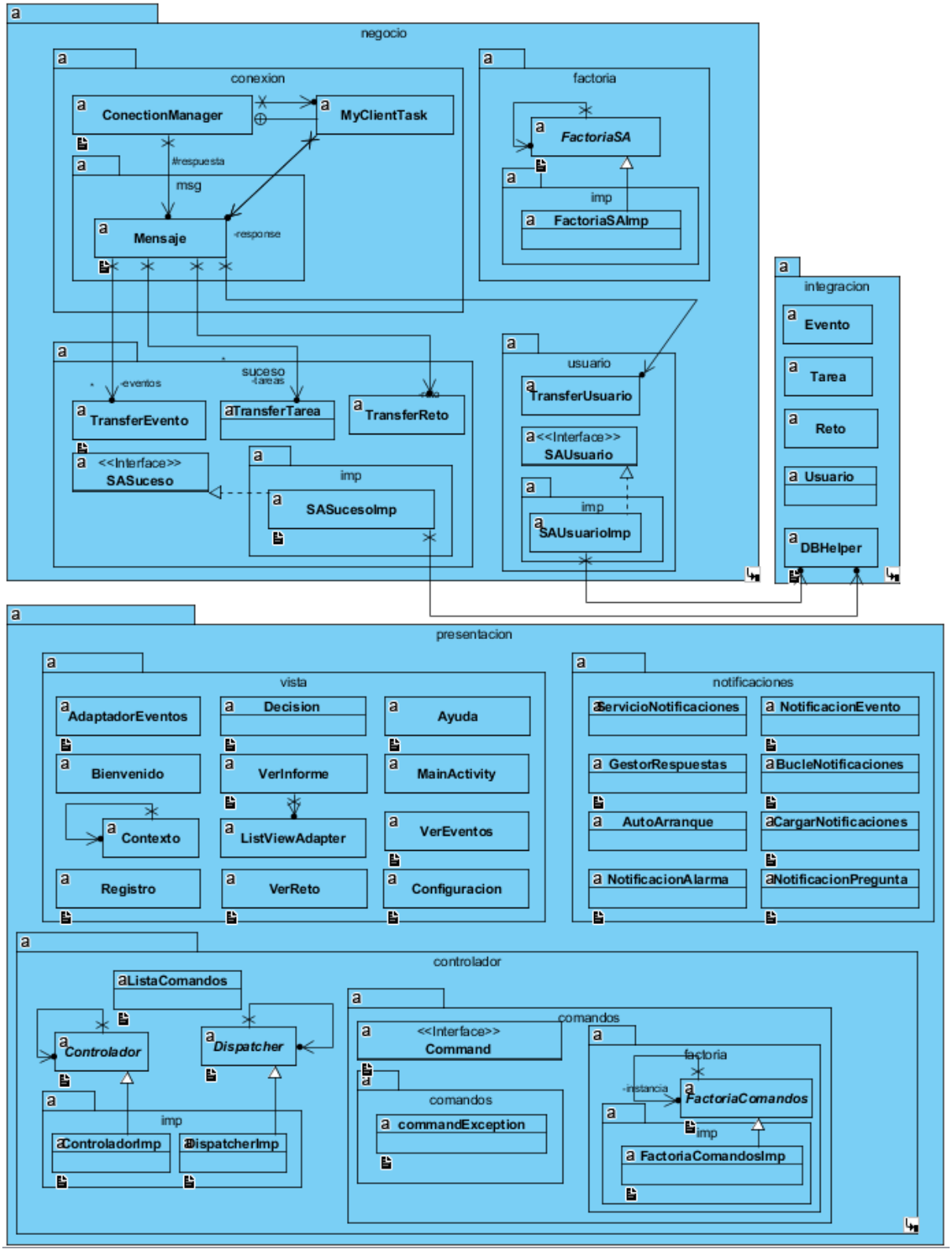
[13] ANDROID DEVELOPERS. Serializable. Disponible en: http://developer.android.com/intl/es/reference/java/io/Serializable.html

[14] FUNDACIÓN ORANGE. Convocatoria Soluciones Digitales - para desarrollo de proyectos. Disponible en: http://www.fundacionorange.es/junto-al-autismo/convocatorias-de-proyectos/convocatoria-soluciones-digitales/

ANEXO A: Diagramas de clase

A.1 Aplicación usuario

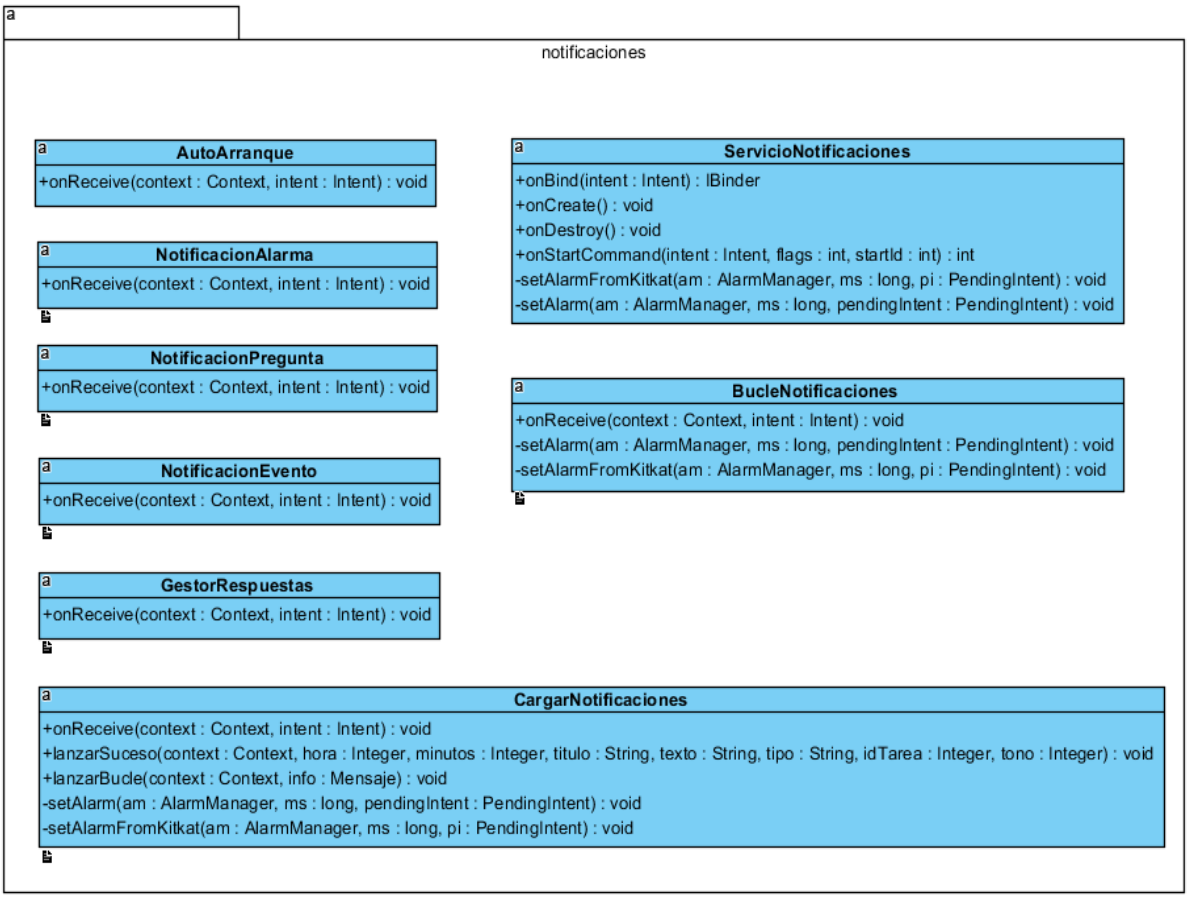
Primero se muestran un diagrama general de la aplicación para que se vea la relación entre los paquetes principales. A continuación, se muestran con detalle los diagramas de clase del paquete de presentación. Después los diagramas del paquete de negocio. Por último, los del paquete de integración.


negocio
conexion
ConectionManager
MyClientTask
#respuesta
msg
Mensaje
-response
factoria
FactoriaSA
imp
FactoriaSAImp
integracion
Evento
Tarea
Reto
Usuario
DBHelper
-eventos
suceso
-tareas
TransferEvento
TransferTarea
TransferReto
<<Interface>>
SASuceso
imp
SASucesoImp
usuario
TransferUsuario
<<Interface>>
SAUsuario
imp
SAUsuarioImp
presentacion
vista
AdaptadorEventos
Decision
Ayuda
Bienvenido
VerInforme
MainActivity
Contexto
ListViewAdapter
VerEventos
Registro
VerReto
Configuracion
notificaciones
ServicioNotificaciones
NotificacionEvento
GestorRespuestas
BucleNotificaciones
AutoArranque
CargarNotificaciones
NotificacionAlarma
NotificacionPregunta
controlador
ListaComandos
Controlador
Dispatcher
imp
ControladorImp
DispatcherImp
comandos
<<Interface>>
Command
comandos
commandException
factoria
-instancia
FactoriaComandos
imp
FactoriaComandosImp

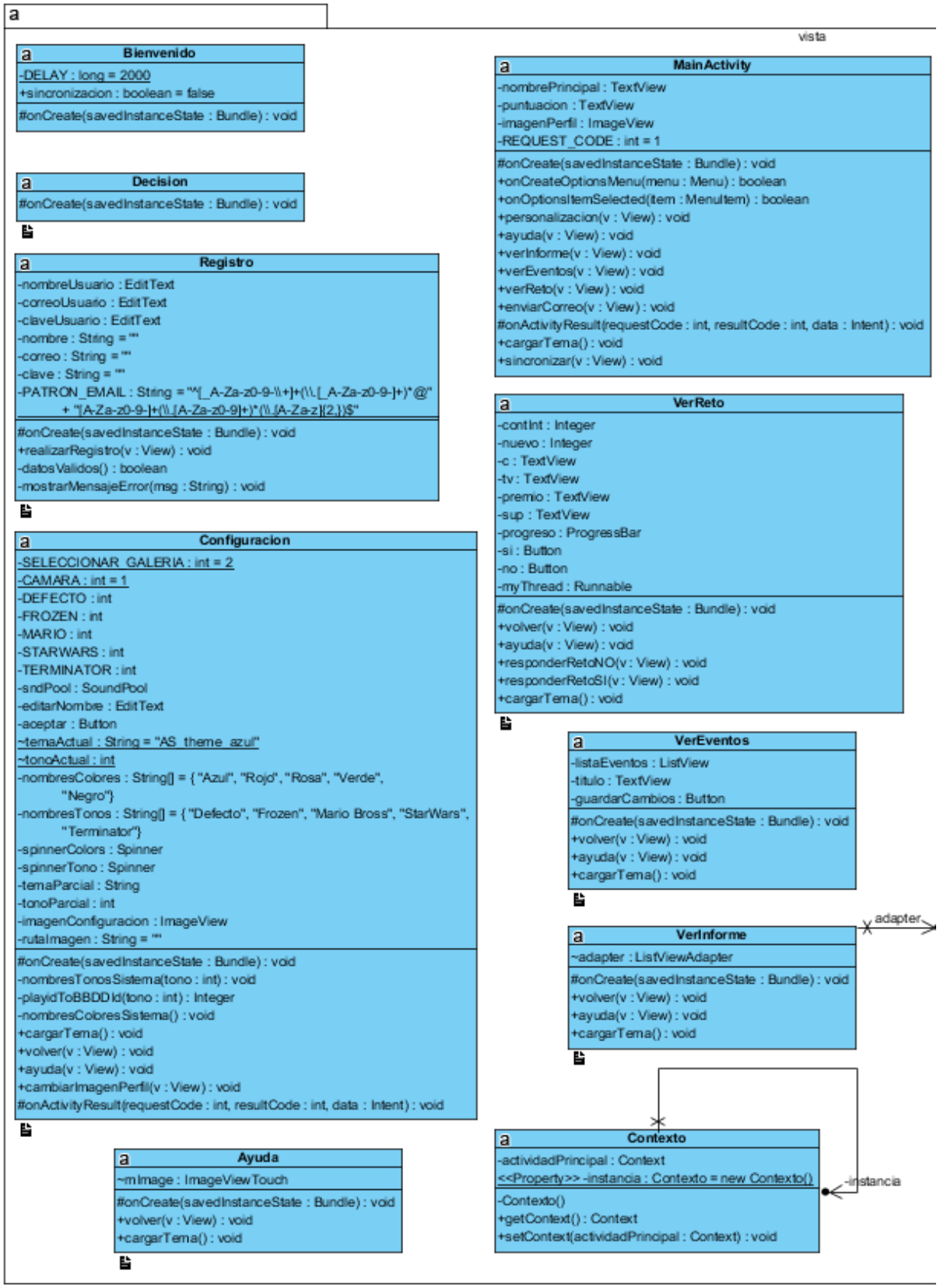
A.1.1 Presentación

Este paquete está subdividido en tres: notificaciones, vista y controlador. Todas las clases que se encuentran en este paquete tienen relación con la interfaz de la *aplicación usuario*.

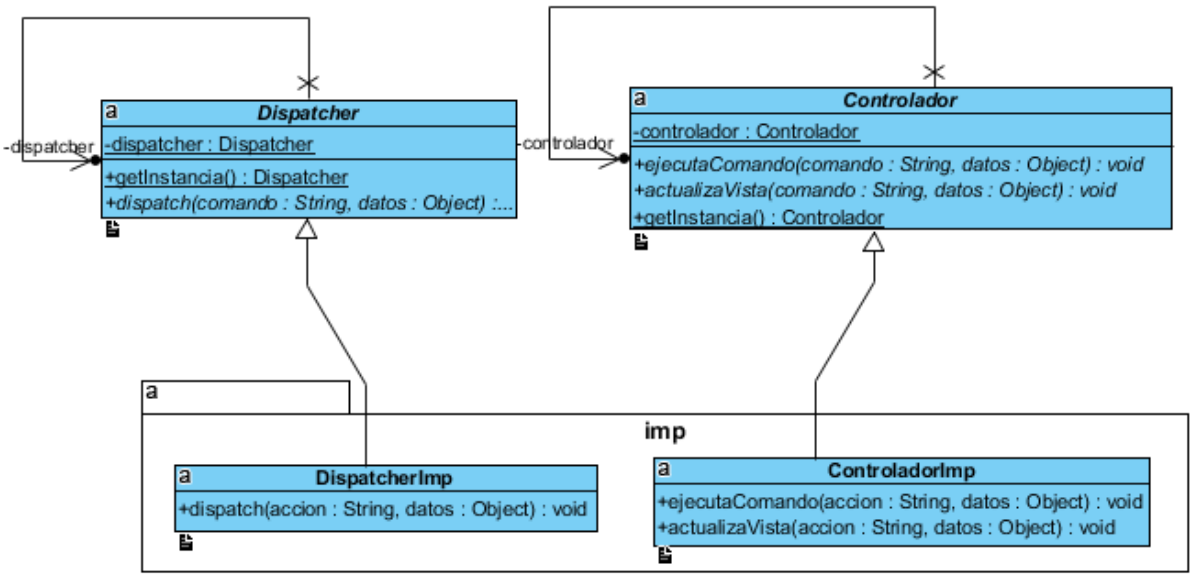
- **Notificaciones**: Son aquellas clases relacionadas con el sistema de notificaciones de la aplicación. ServicioNotificaciones es un servicio de Android, la clase AutoArranque es una actividad y todas las demás extienden de BroadcastReceiver.

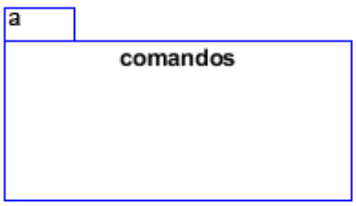


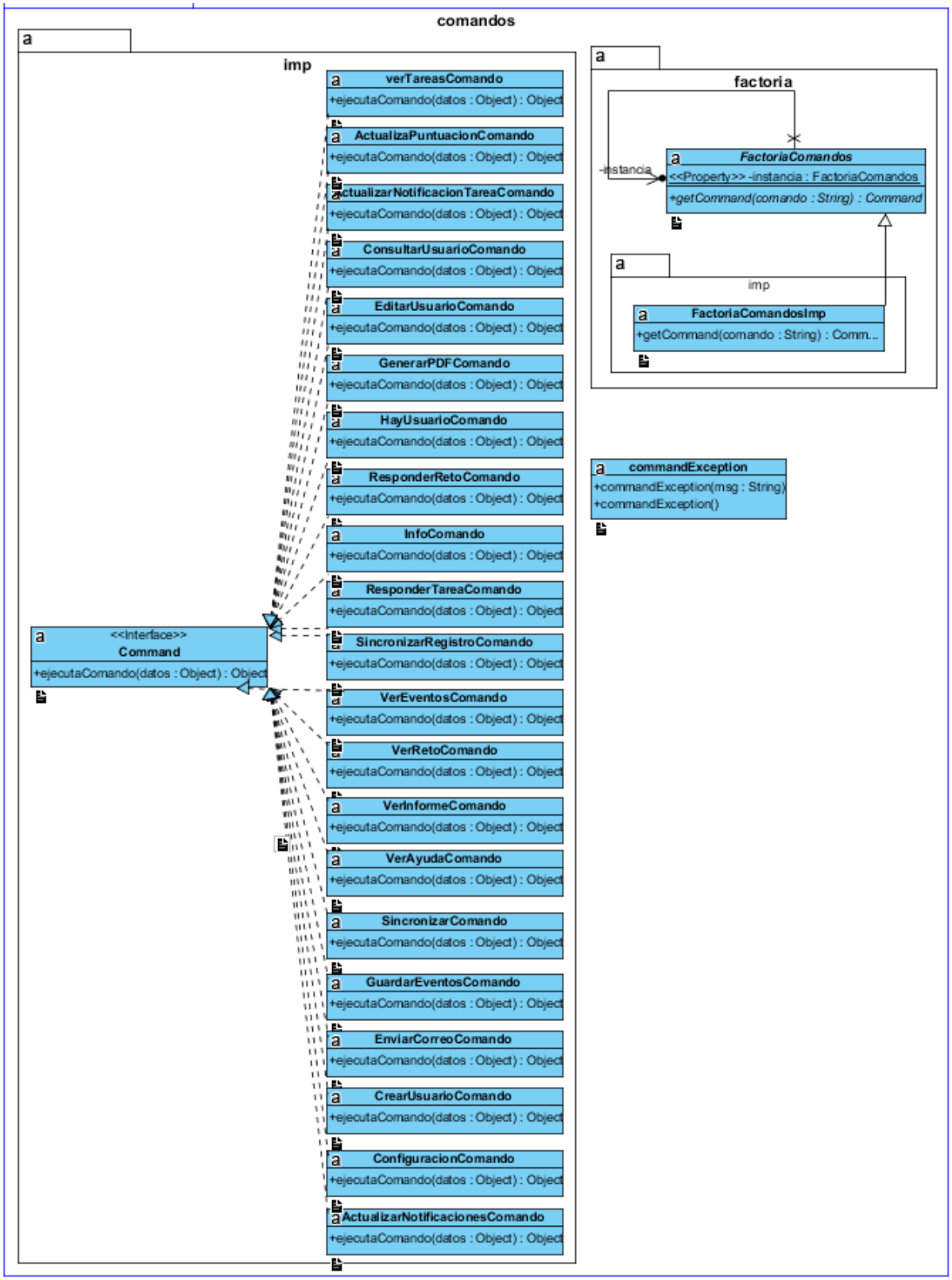
➢ **Vista**: Engloba las clases que están relacionadas con un layout (.xml). También se encuentra la clase Context que es un singleton tal y como se explico en el capítulo 6.



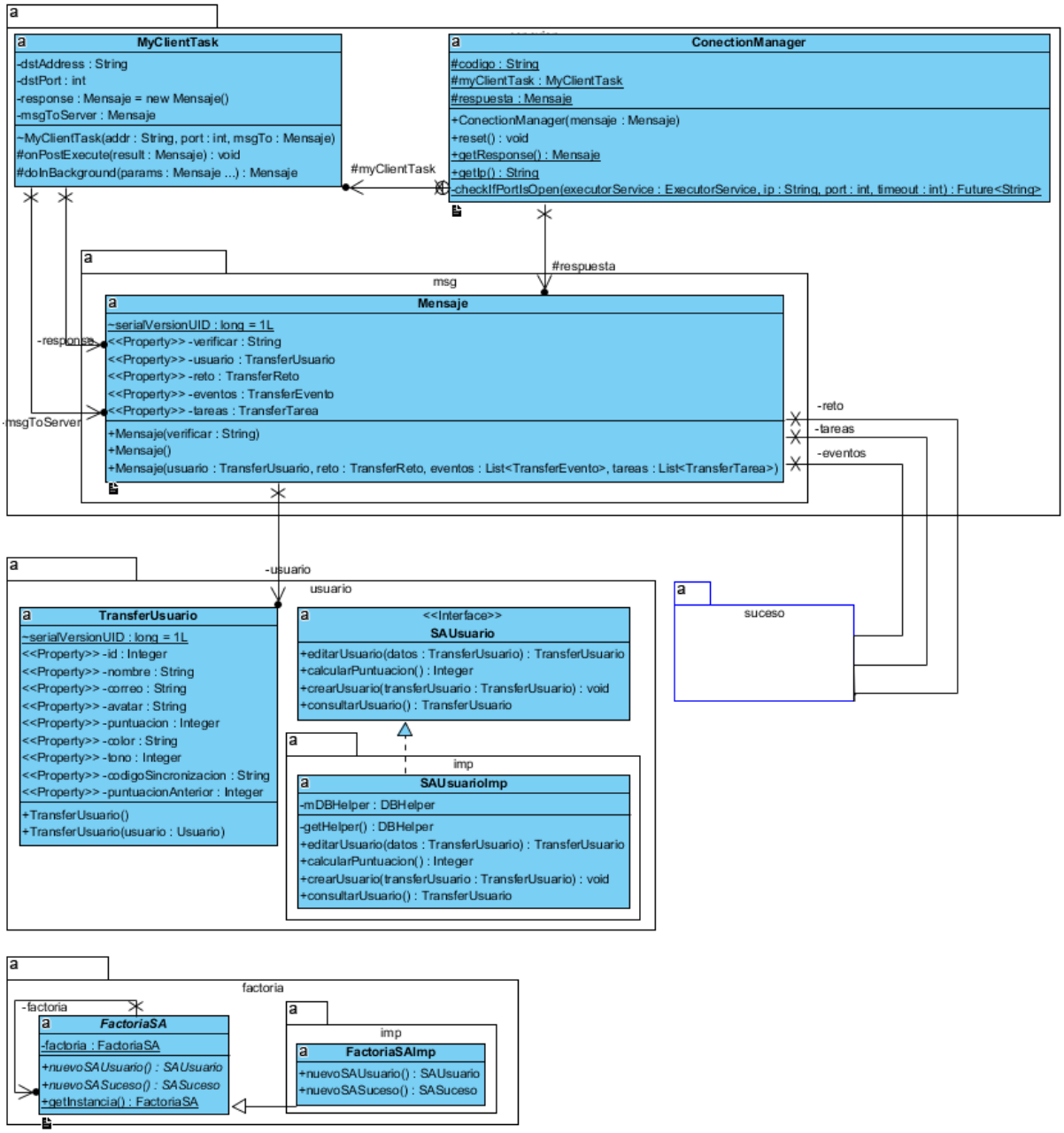
➢ **Controlador**: En este paquete se recogen las clases que forman el controlador debido a los patrones de ingeniería del software que se explicaron en el capítulo 5. Además del controlador y del dispatcher también se encuentra la interfaz Command con todas las que heredan de ella implementando distintos comandos.

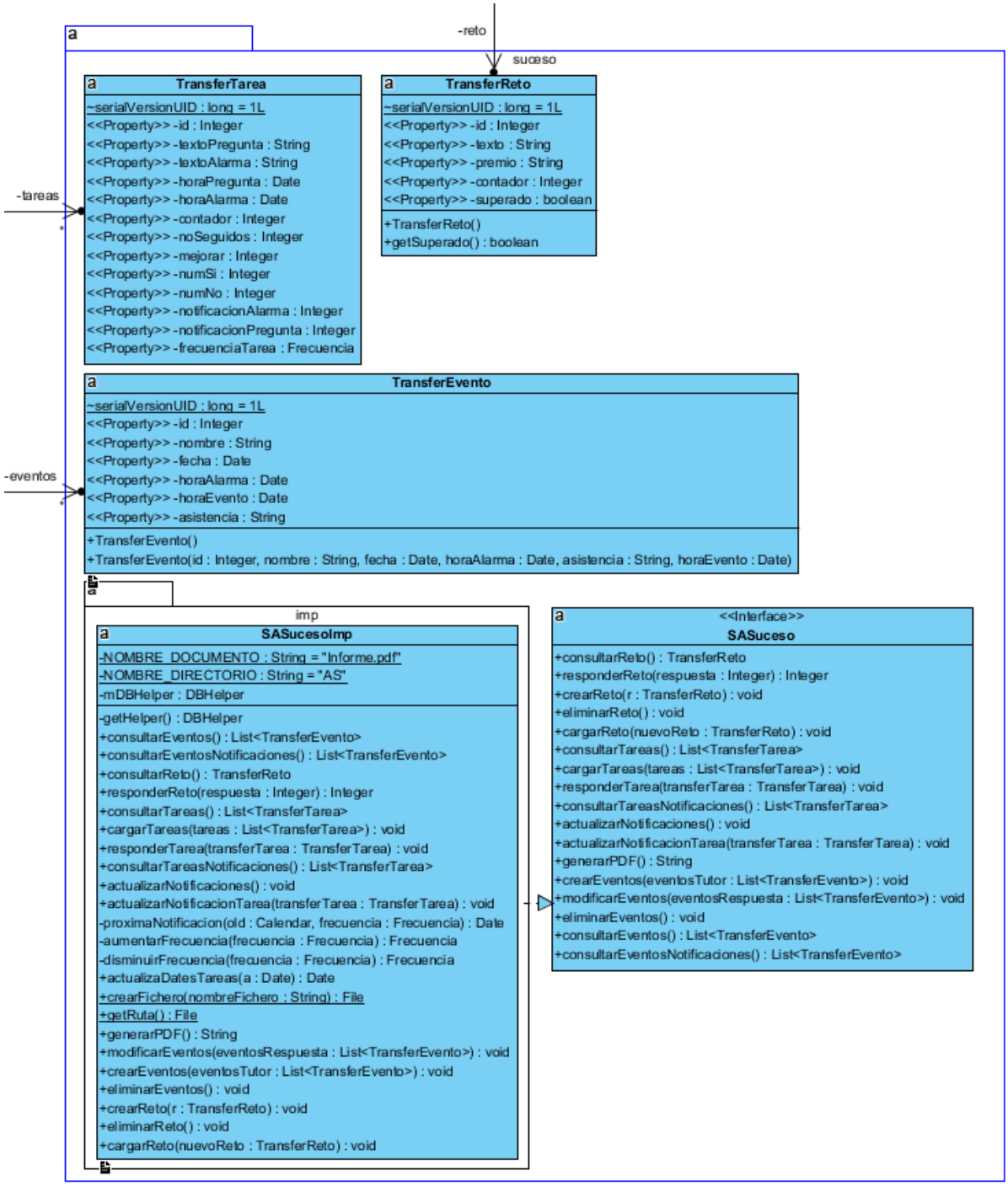




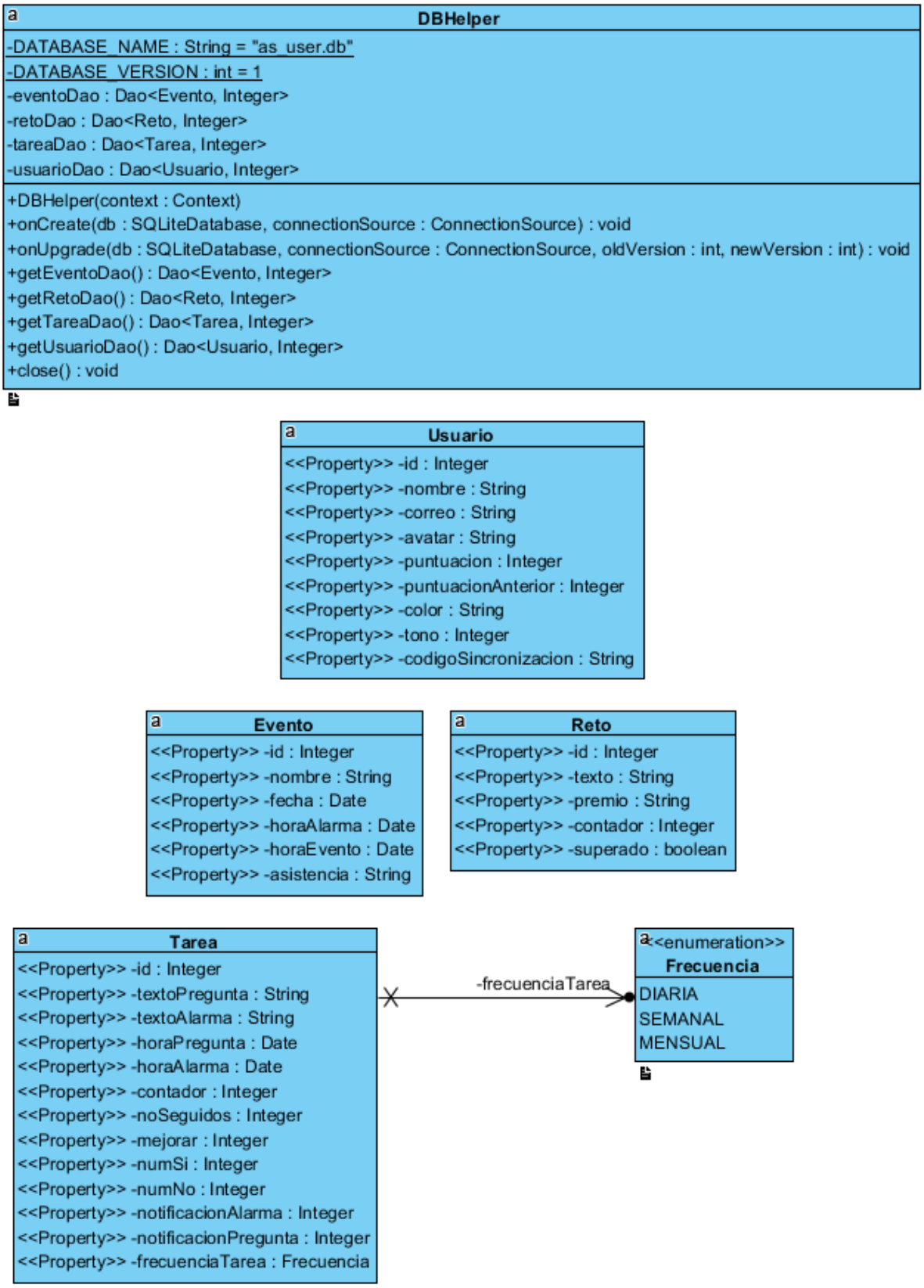

comandos
imp
verTareasComando
+ejecutaComando(datos : Object) : Object
ActualizaPuntuacionComando
+ejecutaComando(datos : Object) : Object
ActualizarNotificacionTareaComando
+ejecutaComando(datos : Object) : Object
ConsultarUsuarioComando
+ejecutaComando(datos : Object) : Object
EditarUsuarioComando
+ejecutaComando(datos : Object) : Object
GenerarPDFComando
+ejecutaComando(datos : Object) : Object
HayUsuarioComando
+ejecutaComando(datos : Object) : Object
ResponderRetoComando
+ejecutaComando(datos : Object) : Object
InfoComando
+ejecutaComando(datos : Object) : Object
ResponderTareaComando
+ejecutaComando(datos : Object) : Object
SincronizarRegistroComando
+ejecutaComando(datos : Object) : Object
VerEventosComando
+ejecutaComando(datos : Object) : Object
VerRetoComando
+ejecutaComando(datos : Object) : Object
VerInformeComando
+ejecutaComando(datos : Object) : Object
VerAyudaComando
+ejecutaComando(datos : Object) : Object
SincronizarComando
+ejecutaComando(datos : Object) : Object
GuardarEventosComando
+ejecutaComando(datos : Object) : Object
EnviarCorreoComando
+ejecutaComando(datos : Object) : Object
CrearUsuarioComando
+ejecutaComando(datos : Object) : Object
ConfiguracionComando
+ejecutaComando(datos : Object) : Object
ActualizarNotificacionesComando
+ejecutaComando(datos : Object) : Object
<<Interface>>
Command
+ejecutaComando(datos : Object) : Object
factoria
FactoriaComandos
<<Property>> -instancia : FactoriaComandos
+getCommand(comando : String) : Command
-instancia
imp
FactoriaComandosImp
+getCommand(comando : String) : Comm...
commandException
+commandException(msg : String)
+commandException()

A.1.2 Negocio


MyClientTask
-dstAddress : String
-dstPort : int
-response : Mensaje = new Mensaje()
-msgToServer : Mensaje
~MyClientTask(addr : String, port : int, msgTo : Mensaje)
#onPostExecute(result : Mensaje) : void
#doInBackground(params : Mensaje ...) : Mensaje
#myClientTask
ConectionManager
#codigo : String
#myClientTask : MyClientTask
#respuesta : Mensaje
+ConectionManager(mensaje : Mensaje)
+reset() : void
+getResponse() : Mensaje
+getIp() : String
-checkIfPortIsOpen(executorService : ExecutorService, ip : String, port : int, timeout : int) : Future<String>
#respuesta
msg
Mensaje
~serialVersionUID : long = 1L
<<Property>> -verificar : String
<<Property>> -usuario : TransferUsuario
<<Property>> -reto : TransferReto
<<Property>> -eventos : TransferEvento
<<Property>> -tareas : TransferTarea
+Mensaje(verificar : String)
+Mensaje()
+Mensaje(usuario : TransferUsuario, reto : TransferReto, eventos : List<TransferEvento>, tareas : List<TransferTarea>)
-response
-msgToServer
-reto
-tareas
-eventos
-usuario
usuario
TransferUsuario
~serialVersionUID : long = 1L
<<Property>> -id : Integer
<<Property>> -nombre : String
<<Property>> -correo : String
<<Property>> -avatar : String
<<Property>> -puntuacion : Integer
<<Property>> -color : String
<<Property>> -tono : Integer
<<Property>> -codigoSincronizacion : String
<<Property>> -puntuacionAnterior : Integer
+TransferUsuario()
+TransferUsuario(usuario : Usuario)
<<Interface>>
SAUsuario
+editarUsuario(datos : TransferUsuario) : TransferUsuario
+calcularPuntuacion() : Integer
+crearUsuario(transferUsuario : TransferUsuario) : void
+consultarUsuario() : TransferUsuario
imp
SAUsuarioImp
-mDBHelper : DBHelper
-getHelper() : DBHelper
+editarUsuario(datos : TransferUsuario) : TransferUsuario
+calcularPuntuacion() : Integer
+crearUsuario(transferUsuario : TransferUsuario) : void
+consultarUsuario() : TransferUsuario
suceso
factoria
-factoria
FactoriaSA
-factoria : FactoriaSA
+nuevoSAUsuario() : SAUsuario
+nuevoSASuceso() : SASuceso
+getInstancia() : FactoriaSA
imp
FactoriaSAImp
+nuevoSAUsuario() : SAUsuario
+nuevoSASuceso() : SASuceso

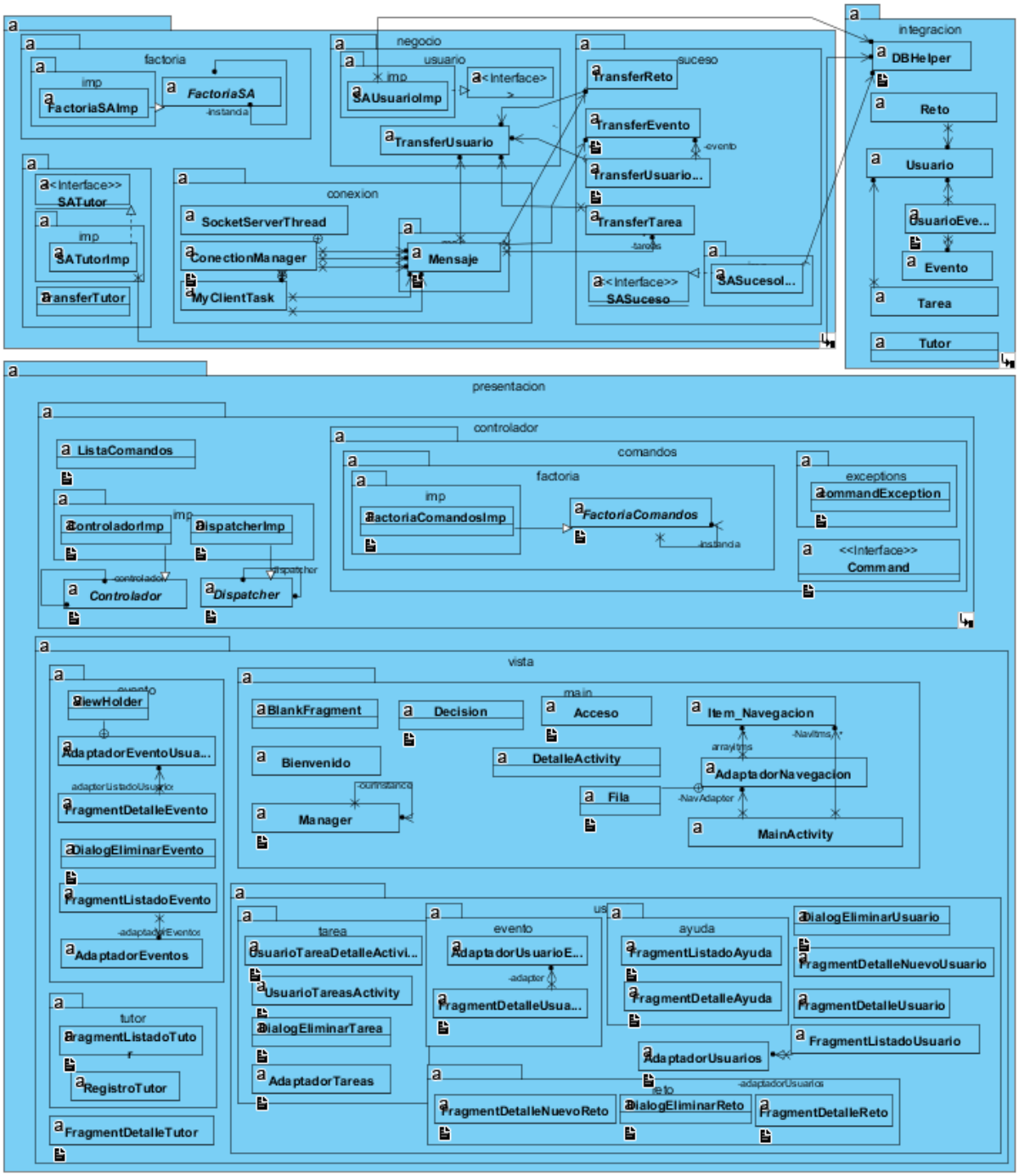

a
-reto
suceso
a TransferTarea
~serialVersionUID : long = 1L
<<Property>> -id : Integer
<<Property>> -textoPregunta : String
<<Property>> -textoAlarma : String
<<Property>> -horaPregunta : Date
<<Property>> -horaAlarma : Date
<<Property>> -contador : Integer
<<Property>> -noSeguidos : Integer
<<Property>> -mejorar : Integer
<<Property>> -numSi : Integer
<<Property>> -numNo : Integer
<<Property>> -notificacionAlarma : Integer
<<Property>> -notificacionPregunta : Integer
<<Property>> -frecuenciaTarea : Frecuencia
-tareas
a TransferReto
~serialVersionUID : long = 1L
<<Property>> -id : Integer
<<Property>> -texto : String
<<Property>> -premio : String
<<Property>> -contador : Integer
<<Property>> -superado : boolean
+TransferReto()
+getSuperado() : boolean
a TransferEvento
~serialVersionUID : long = 1L
<<Property>> -id : Integer
<<Property>> -nombre : String
<<Property>> -fecha : Date
<<Property>> -horaAlarma : Date
<<Property>> -horaEvento : Date
<<Property>> -asistencia : String
-eventos
+TransferEvento()
+TransferEvento(id : Integer, nombre : String, fecha : Date, horaAlarma : Date, asistencia : String, horaEvento : Date)
imp
a SASucesoImp
-NOMBRE_DOCUMENTO : String = "Informe.pdf"
-NOMBRE_DIRECTORIO : String = "AS"
-mDBHelper : DBHelper
-getHelper() : DBHelper
+consultarEventos() : List<TransferEvento>
+consultarEventosNotificaciones() : List<TransferEvento>
+consultarReto() : TransferReto
+responderReto(respuesta : Integer) : Integer
+consultarTareas() : List<TransferTarea>
+cargarTareas(tareas : List<TransferTarea>) : void
+responderTarea(transferTarea : TransferTarea) : void
+consultarTareasNotificaciones() : List<TransferTarea>
+actualizarNotificaciones() : void
+actualizarNotificacionTarea(transferTarea : TransferTarea) : void
-proximaNotificacion(old : Calendar, frecuencia : Frecuencia) : Date
-aumentarFrecuencia(frecuencia : Frecuencia) : Frecuencia
-disminuirFrecuencia(frecuencia : Frecuencia) : Frecuencia
+actualizaDatesTareas(a : Date) : Date
+crearFichero(nombreFichero : String) : File
+getRuta() : File
+generarPDF() : String
+modificarEventos(eventosRespuesta : List<TransferEvento>) : void
+crearEventos(eventosTutor : List<TransferEvento>) : void
+eliminarEventos() : void
+crearReto(r : TransferReto) : void
+eliminarReto() : void
+cargarReto(nuevoReto : TransferReto) : void
a <<Interface>>
SASuceso
+consultarReto() : TransferReto
+responderReto(respuesta : Integer) : Integer
+crearReto(r : TransferReto) : void
+eliminarReto() : void
+cargarReto(nuevoReto : TransferReto) : void
+consultarTareas() : List<TransferTarea>
+cargarTareas(tareas : List<TransferTarea>) : void
+responderTarea(transferTarea : TransferTarea) : void
+consultarTareasNotificaciones() : List<TransferTarea>
+actualizarNotificaciones() : void
+actualizarNotificacionTarea(transferTarea : TransferTarea) : void
+generarPDF() : String
+crearEventos(eventosTutor : List<TransferEvento>) : void
+modificarEventos(eventosRespuesta : List<TransferEvento>) : void
+eliminarEventos() : void
+consultarEventos() : List<TransferEvento>
+consultarEventosNotificaciones() : List<TransferEvento>

A.1.3 Integración

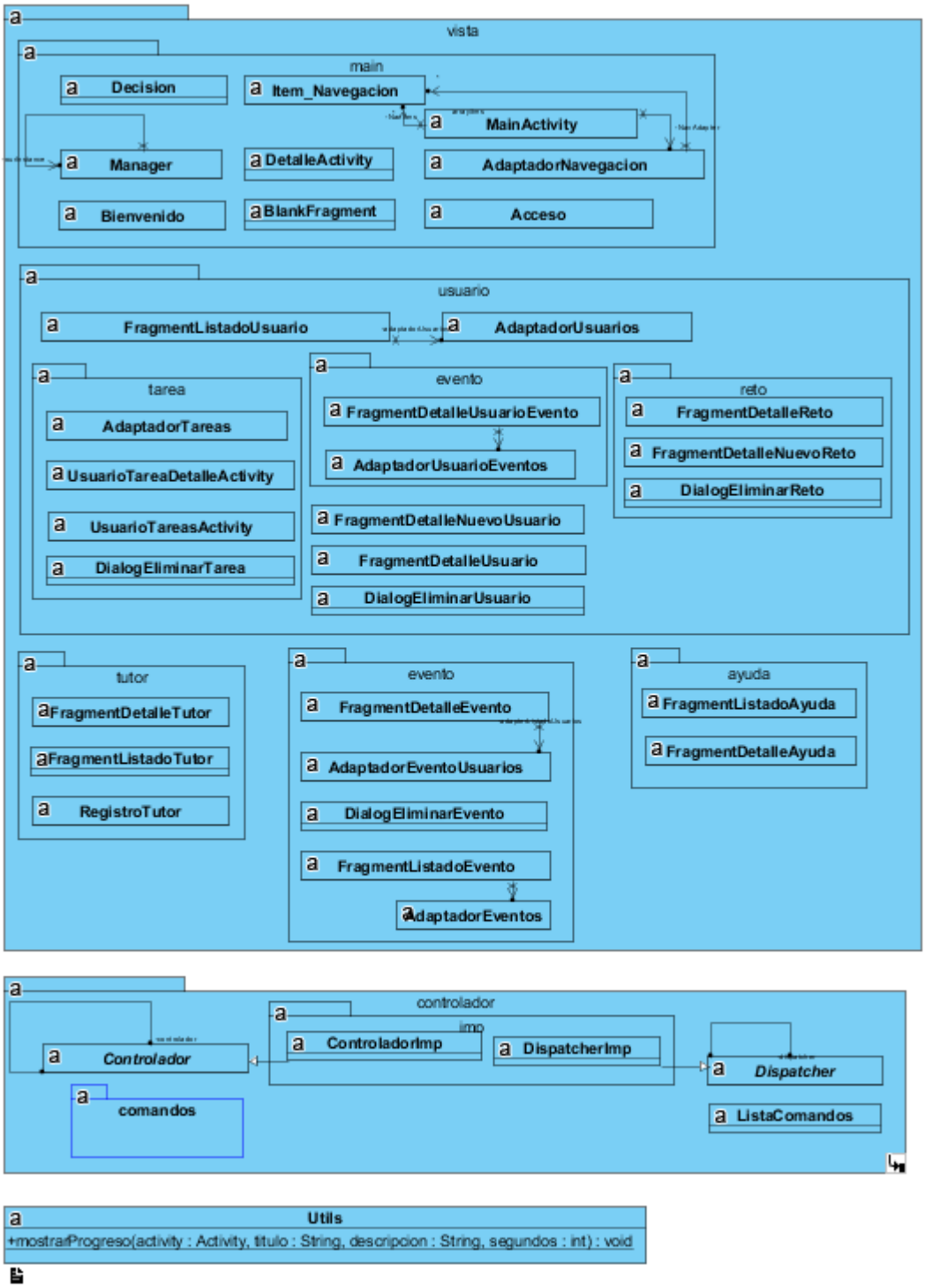


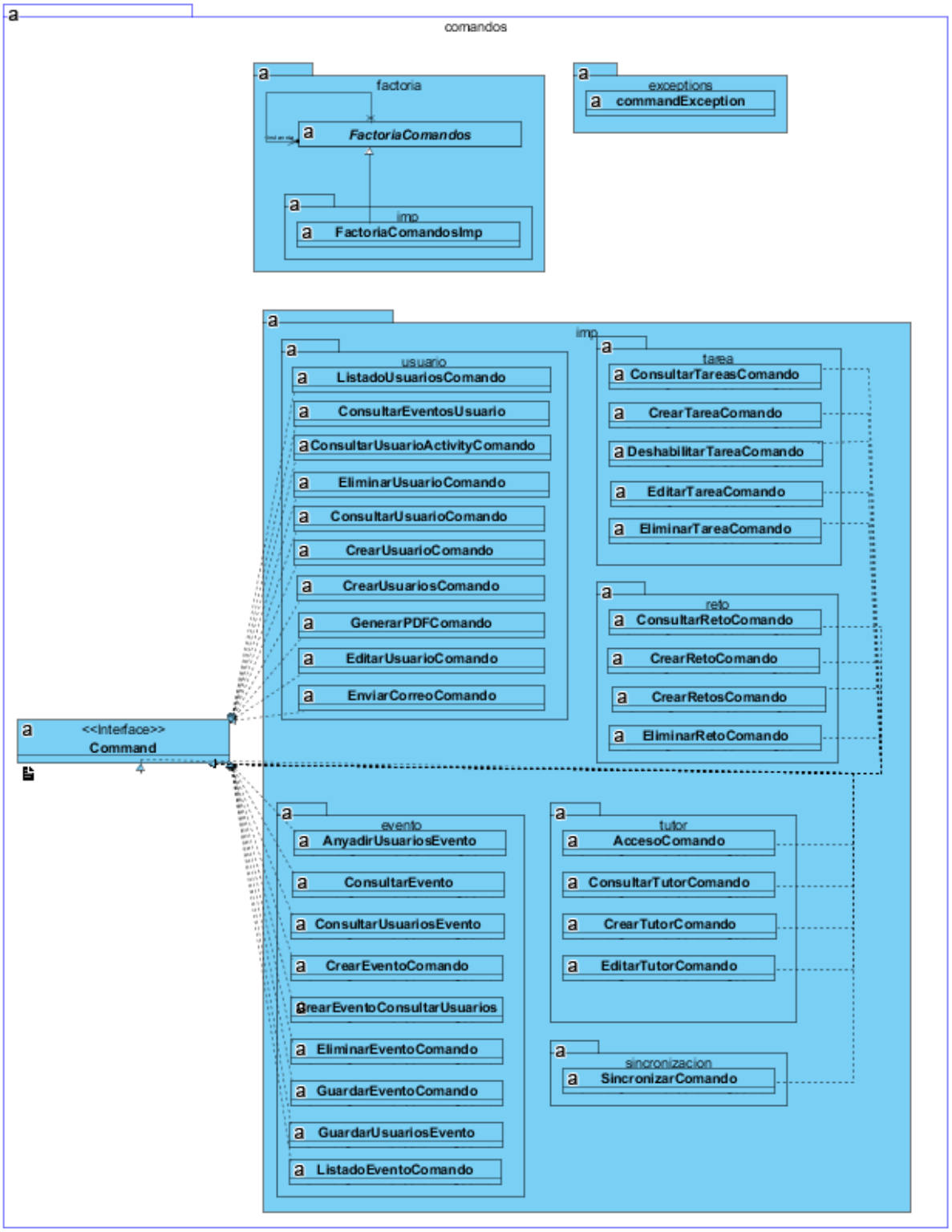
A.2 Aplicación tutor

Primero se muestran un diagrama general de la aplicación para que se vea la relación entre los paquetes principales. A continuación, se detallan los diagramas de clase de cada paquete.

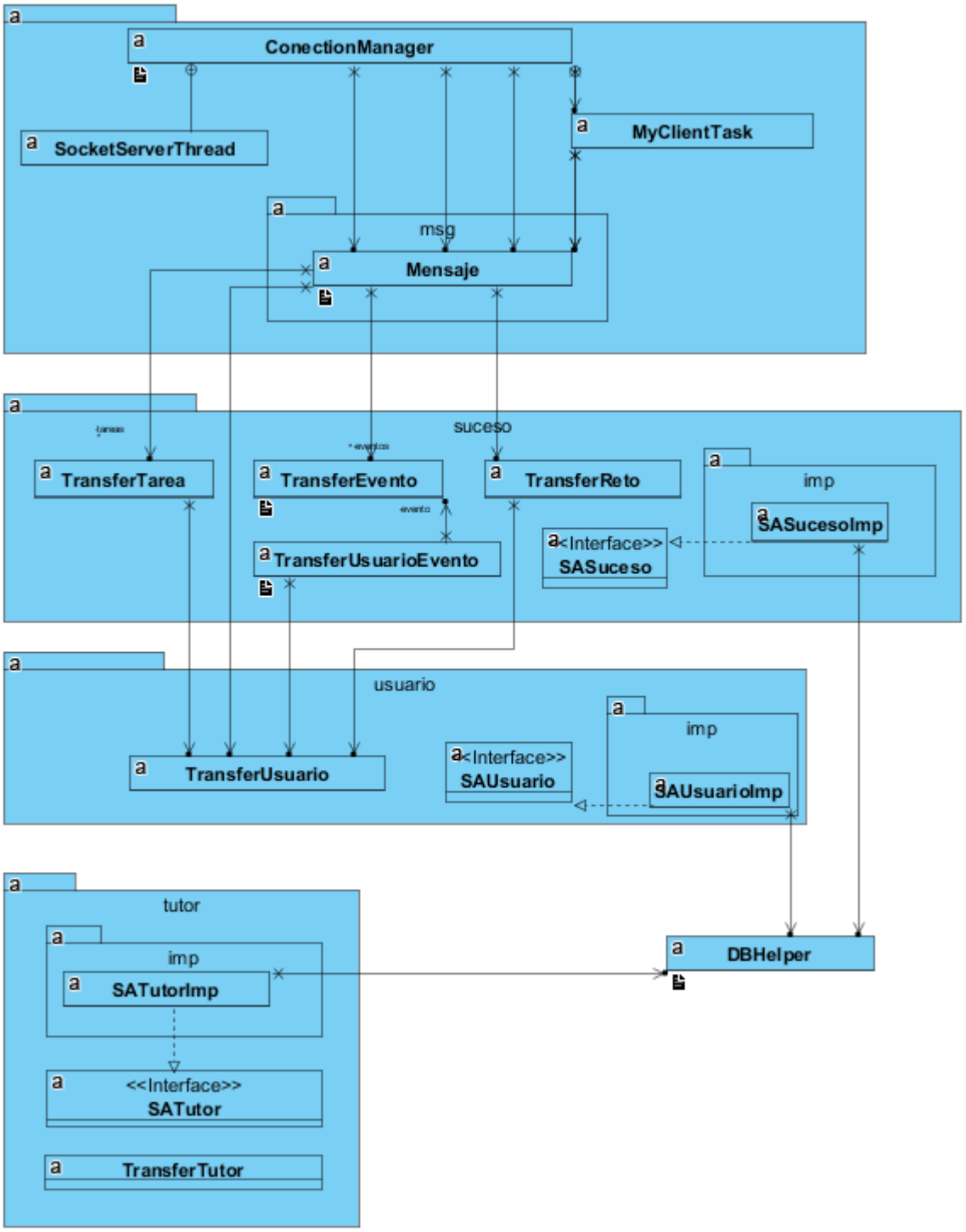


A.2.1 Presentación

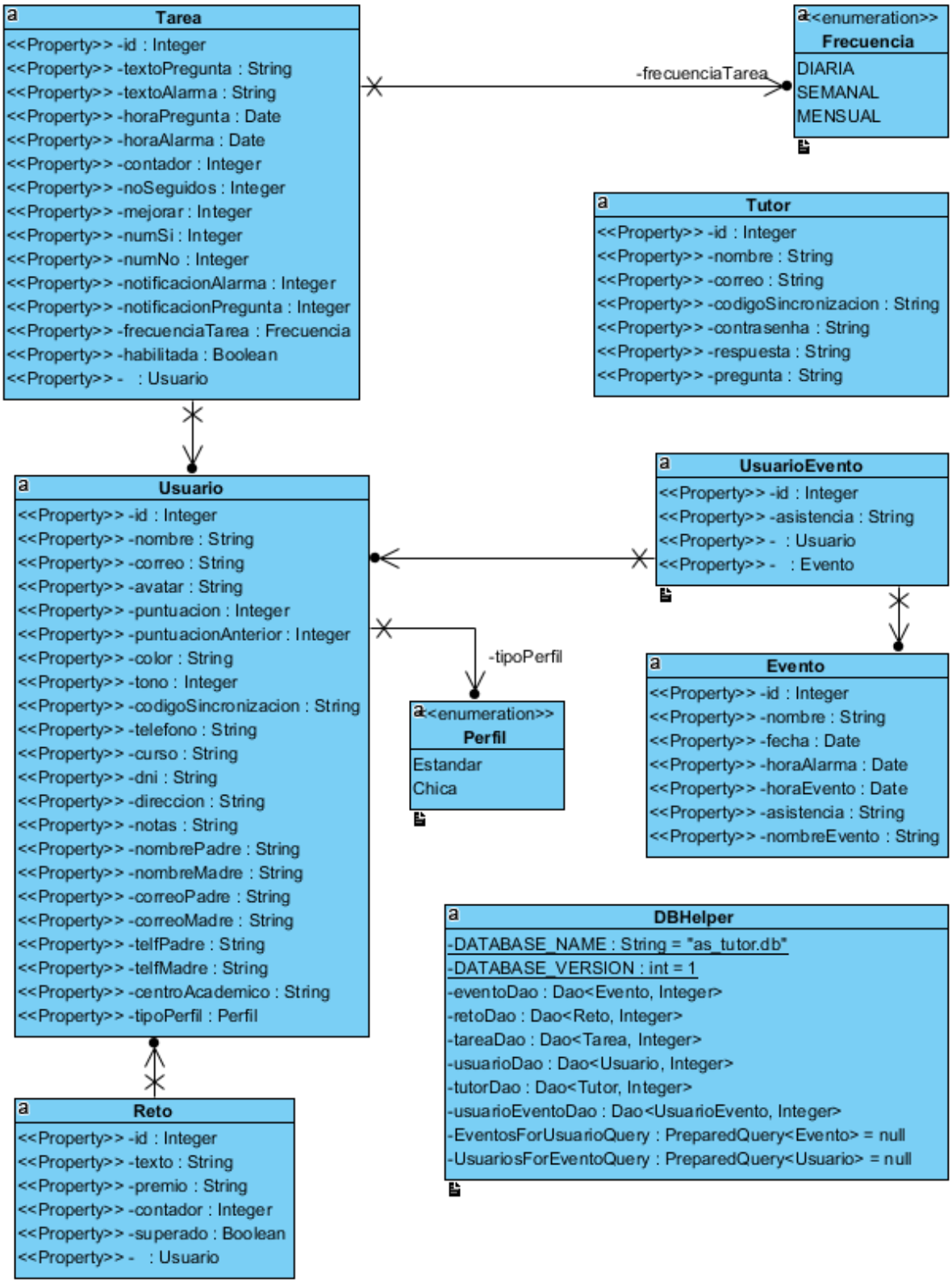



comandos
factoria
FactoriaComandos
imp
FactoriaComandosImp
exceptions
commandException
imp
usuario
ListadoUsuariosComando
ConsultarEventosUsuario
ConsultarUsuarioActivityComando
EliminarUsuarioComando
ConsultarUsuarioComando
CrearUsuarioComando
CrearUsuariosComando
GenerarPDFComando
EditarUsuarioComando
EnviarCorreoComando
tarea
ConsultarTareasComando
CrearTareaComando
DeshabilitarTareaComando
EditarTareaComando
EliminarTareaComando
reto
ConsultarRetoComando
CrearRetoComando
CrearRetosComando
EliminarRetoComando
<<Interface>>
Command
evento
AnyadirUsuariosEvento
ConsultarEvento
ConsultarUsuariosEvento
CrearEventoComando
CrearEventoConsultarUsuarios
EliminarEventoComando
GuardarEventoComando
GuardarUsuariosEvento
ListadoEventoComando
tutor
AccesoComando
ConsultarTutorComando
CrearTutorComando
EditarTutorComando
sincronizacion
SincronizarComando

A.2.2 Negocio


ConectionManager
SocketServerThread
MyClientTask
msg
Mensaje
suceso
TransferTarea
TransferEvento
TransferReto
TransferUsuarioEvento
imp
SASucesoImp
<<Interface>>
SASuceso
usuario
TransferUsuario
<<Interface>>
SAUsuario
imp
SAUsuarioImp
tutor
imp
SATutorImp
<<Interface>>
SATutor
TransferTutor
DBHelper

A.2.3 Integración



ANEXO B: Cronograma de la planificación