GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

APLICACIÓN MÓVIL DE GESTIÓN DE EVENTOS GRUPALES CON TOMAS DE DECISIÓN

Autor: Juan Sebastián de Benito Cháfer

Director: Miguel Manuel Martín Lopo

Madrid

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Aplicación móvil de gestión de eventos grupales con tomas de decisión

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2019/20 es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: Juan Sebastián de Benito Cháfer Fecha: 11 / 07 / 2020

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Miguel Manuel Martín Lopo Fecha: 11 / 07 / 2020

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

APLICACIÓN MÓVIL DE GESTIÓN DE EVENTOS GRUPALES CON TOMAS DE DECISIÓN

Autor: Juan Sebastían de Benito Cháfer

Director: Miguel Manuel Martín Lopo

Madrid

Agradecimientos

Gracias a todas aquellos familiares y amigos que me han acompañado a lo largo de estos cuatro años, Querría hacer una mención especial a Rocky, por todas las horas pasadas en el laboratorio; y a Nacho, por todas las pasadas en otros sitios.

Ya no sabría decir que fue de aquel chaval de 17 años que entró por primera vez por las puertas de ICAI en 2016. Supongo que ahora tiene 21.

APLICACIÓN MÓVIL DE GESTIÓN DE EVENTOS GRUPALES CON TOMAS DE DECISIÓN

Autor: de Benito Cháfer, Juan Sebastián.
Director: Martín Lopo, Miguel Manuel.
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

El proyecto desarrollado modela un sistema que permita a un usuario crear un evento y compartirlo para que los invitados puedan apreciar sus características y realizar cambios en las mismas. Para ello, serán necesarios dos elementos esenciales: una aplicación móvil multiplataforma como interfaz de usuario y un servidor como gestor principal de la información.

Palabras clave: *scheduler*, aplicación, multiplataforma, fechas, votación, nube

1. Introducción

La gestión del tiempo siempre ha sido un tema de preocupación para el ser humano, por lo que no es de extrañar que diversas herramientas, como calendarios o agendas, fueran apareciendo gradualmente a lo largo de la historia. Con la llegada de la Era Digital, sus homólogos electrónicos comenzaron a cobrar importancia en el ámbito laboral y privado debido a las nuevas funcionalidades disponibles, como la sincronización automática entre dispositivos (móviles, tabletas, ordenadores…) o la posibilidad de compartir eventos con otros usuarios a través de Internet.

En este ámbito surgieron dos aplicaciones muy empleadas en la actualidad: los calendarios electrónicos, como Google Calendar o Outlook, y los *schedulers*, como Doodle o Bash. Los primeros, al igual que sus homólogos analógicos, sirven para anotar eventos y citas en el tiempo.

Por su parte, los *schedulers* se emplean para organizar encuentros entre varias personas, comparando la disponibilidad de los participantes a través de un sistema de propuesta-respuesta —el administrador propone una serie de fechas y los invitados seleccionan las que mejor les convienen—. Pese a que son herramientas muy útiles, su uso no suele ser especialmente intuitivo, además de que presentan una limitación funcional importante: los invitados al evento no tienen los mecanismos necesarios para proponer sus propias fechas, lo que limita mucho su rango de acción.

2. Definición del proyecto

El objetivo de este proyecto es crear una aplicación móvil multiplataforma capaz de integrar de forma efectiva la visual e intuitiva interfaz de los calendarios electrónicos con el amplio abanico de funcionalidades que ofrecen de los *schedulers*. Como novedad, la aplicación permitirá a los invitados proponer sus propias fechas, en lugar de limitarse

al voto. Desde el punto de vista del servidor, se estudiarán dos posibles soluciones: emplear un sistema dedicado o utilizar los servicios en la nube de un tercero, en particular los de la API de Google Calendar.

3. Descripción de la arquitectura

La arquitectura del sistema propuesto está formada por cuatro elementos: la aplicación, el almacenamiento local, el servidor y la base de datos. Su esquema aparece a continuación.

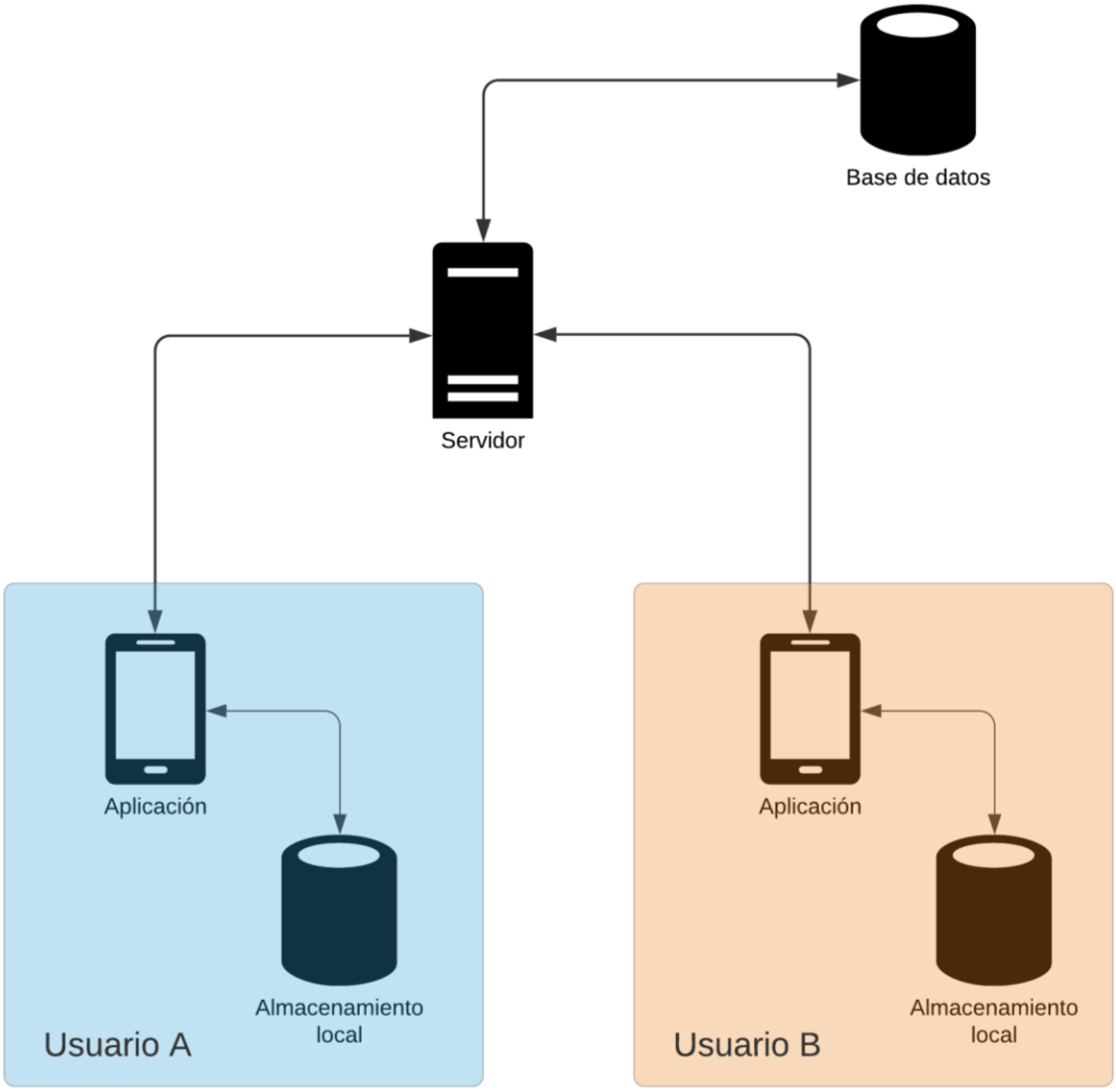

Ilustración 1. Arquitectura del sistema

En primer lugar, la **aplicación** hace las funciones de interfaz de usuario, proveyendo al administrador y sus invitados de los datos y herramientas necesarios para crear y modificar los eventos. Además, es el elemento por el que se accede al sistema

proposición y voto de las fechas. Como se ha mencionado, para su diseño se ha empleado una interfaz sencilla e intuitiva, similar a las empleadas en otros *schedulers* como Doodle o Bash, pero con ciertas características de los calendarios inteligentes. Como es lógico, es el elemento encargado de dialogar con el servidor, ya sea para validar al usuario o para actualizar la información de sus eventos.

Como una extensión de la aplicación nos encontramos el **almacenamiento local**, que se encarga de guardar de forma persistente la información de los eventos del usuario, además de sus credenciales, cuando la aplicación no se encuentra en ejecución. Exceptuando las credenciales, se trata de un elemento prescindible, ya que la aplicación podría obtener todos los parámetros relativos a los eventos al iniciarse y no requerir de memoria persistente. Pese a ello, un sistema de almacenamiento en el dispositivo podría mejorar la experiencia de usuario al permitir el uso de la aplicación offline.

Por su parte, el **servidor** es el eje central de la aplicación. Es el encargado de recibir y procesar la información de los clientes, además de ser el único acceso a la base de datos. Debido a que no sólo almacena toda la información sobre los eventos, sino también sobre los usuarios, este elemento es el que mayor seguridad requiere. El análisis de datos es otra de sus funciones, la cual permite comprender las necesidades de los usuarios de cara a la aplicación.

Para finalizar, la **base de datos** gestiona el guardado de datos de todos los eventos, además de las credenciales de los usuarios. Su sincronización con las aplicaciones y sus respectivos almacenamientos locales (si los hubiera) es crucial, ya que los usuarios no deberían ver información excesivamente desactualizada.

4. Resultados

Se ha conseguido diseñar una aplicación móvil multiplataforma a través de Xamarin.Forms, un marco de interfaz de código abierto, el cual permite compilar aplicaciones de forma nativa para Android, iOS y Windows 10 a través de un único código compartido [1], que en este caso ha sido C#. En la práctica, ha sido únicamente probada en diferentes versiones Android y Windows 10, pero no es iOS por la falta de medios.

Actualmente, la aplicación permite crear y modificar los eventos a través de una interfaz de usuario sencilla e intuitiva, pero no permite compartir eventos con otros usuarios. Pese a ello, el sistema de proposición y voto de fechas se ha desarrollado y probado de forma satisfactoria. A continuación, se muestran unas capturas de la interfaz de la aplicación en su versión Android, donde se pueden apreciar fácilmente los elementos de calendario electrónico introducidos.

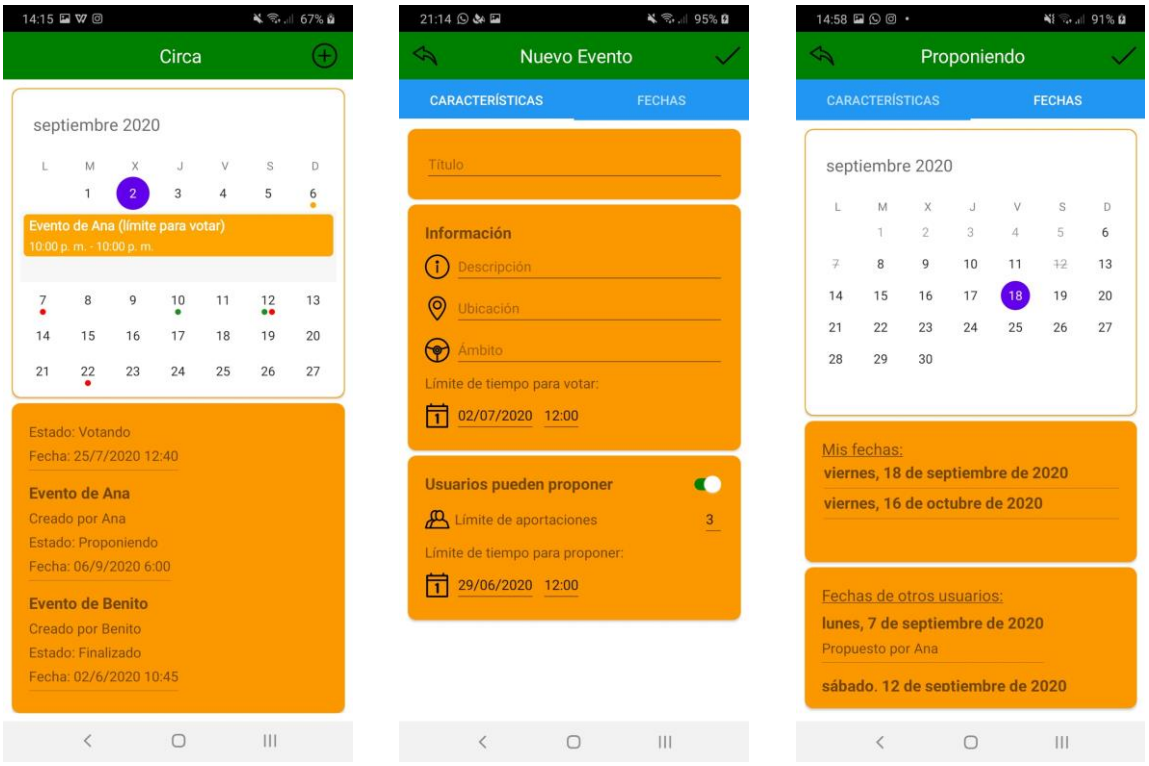

Ilustración 2. Página principal (izquierda) y de gestión de evento (medio, derecha), en su versión Android

Pese a que no ha podido llevarse a cabo ningún avance práctico sobre el lado del servidor, se han estudiado diversas opciones de inicio de sesión y almacenamiento en la nube. Sobre este último punto se ha desarrollado un modelo detallado de cómo podría emplearse la API de Google Calendar, de tal manera que todos los parámetros de los eventos y su gestión pasaran a cargo de los sistemas de la compañía norteamericana.

5. Conclusiones

La aplicación móvil ha conseguido de forma efectiva la integración de la interfaz de los calendarios electrónicos con la funcionalidad de los *schedulers*. Su interfaz es intuitiva y fácil de usar, ya sea en su versión móvil (Android, iOS) o de escritorio (Windows 10). En cuanto a su funcionalidad, se ha conseguido introducir de forma efectiva la posibilidad de que los invitados puedan proponer sus propias opciones, en lugar del sistema tradicional, que se ciñe a las del administrador.

Debido a una infravaloración del tiempo, ciertas capacidades prometidas no han podido llevarse a cabo de forma efectiva, especialmente por el lado del servidor. Pese a ello, la investigación realizada sobre el almacenamiento en la nube revela no sólo que es posible no depender de un servidor y base de datos propios, sino que puede ser recomendable en

ciertos casos. En esta línea, se ha estudiado la integración del sistema con los servicios de Google Calendar, que podría resultar un buen punto de partida para trabajos ulteriores.

6. Referencias

[1] Microsoft Docs, «What is Xamarin? | .NET», *2020*, 2020. [En línea]. Disponible en: https://dotnet.microsoft.com/learn/xamarin/what-is-xamarin. [Accedido: 22-jun-2020].

MOBILE APPLICATION OF MANAGEMENT OF GROUP EVENTS WITH DECISION-MAKING

Author: de Benito Cháfer, Juan Sebastián.
Supervisor: Martín Lopo, Miguel Manuel.
Collaborating Entity: ICAI – Universidad Pontificia Comillas

ABSTRACT

The developed project models a system that allows a user to create an event and share it so that guests can appreciate its characteristics and make changes to them. For this purpose, two essential elements will be necessary: a multiplatform mobile application as the user interface and a server as the main information manager.

Keywords: scheduler, application, multiplatform, dates, voting, cloud

1. Introduction

Time management has always been a matter of concern for human beings, for which is not surprising that various tools, such as calendars or agendas, gradually began to emerge throughout history. With the arrival of the Digital Age, its electronic counterparts began to gain importance in the workplace and the private due to the new functionalities available, like automatic synchronization between devices (mobile, tablets, computers ...) and sharing events with others users through the Internet, among others.

Two applications that are widely used nowadays emerged in this field: electronic calendars, such as Google Calendar or Outlook, and schedulers, such as Doodle or Bash. The former, like their analog counterparts, serve to record events and appointments over time.

The schedulers, for their part, are used to organize meetings between several people, comparing the availability of the participants through a proposal-response system - the administrator proposes a series of dates and the guests select the ones that best suit them. Despite the fact that they are very useful tools, their use is not usually particularly intuitive, in addition to presenting an important functional limitation: the guests at the event do not have the necessary mechanisms to propose their own dates, which greatly limits their range of action.

2. Definition of the project

The objective of this project is to create a multiplatform mobile application capable of effectively integrating the visual and intuitive interface of electronic calendars with the wide range of functionalities that schedulers offer. As a novelty, the application will allow guests to propose their own dates, instead of just voting them. From the server point of view, two possible solutions will be explored: using a dedicated system or

integrating third-party cloud services, in particular the ones offered by Google Calendar's API.

3. Descripción del modelo/sistema/herramienta

The architecture of the proposed system is made up of four elements: the application, the local storage, the server, and the database. Its outline appears below.

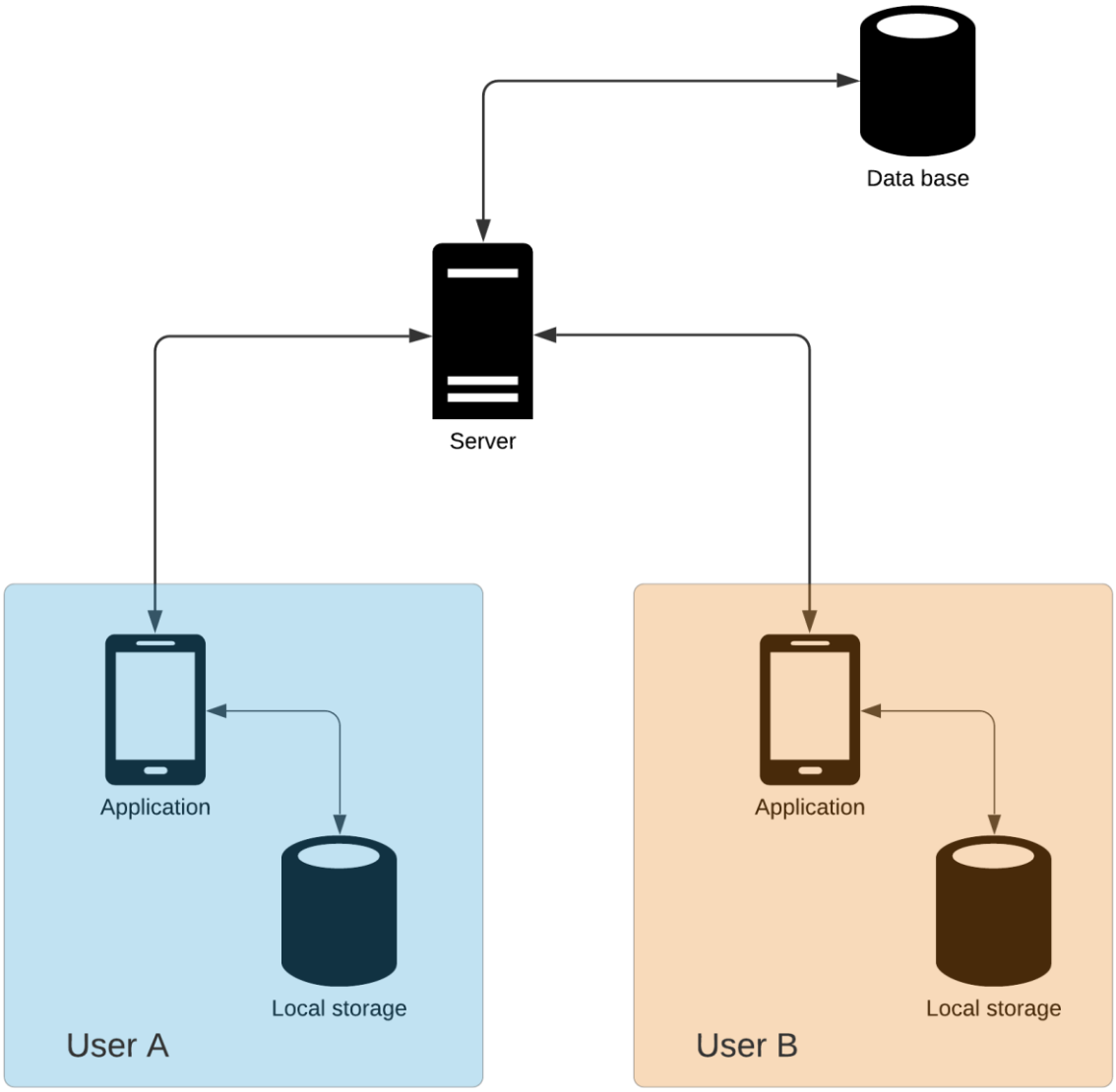

Illustration 1. System architecture

First, the **application** performs the user interface functions, providing the administrator and their guests with the data and tools necessary to create and modify events. In addition, it is the element by which the proposal and vote of dates is accessed. As mentioned, a simple and intuitive interface has been used for its design, like those used in other schedulers such as Doodle or Bash, but with certain characteristics of smart

calendars. As is logical, it is the element in charge of dialoguing with the server, either to validate the user or update the information of its events.

As an extension of the application we find the **local storage**, which is responsible for persistently saving the information of the user's events, in addition to their credentials, when the application is not running. Except for credentials, this is a dispensable item, since the application could obtain all the parameters related to the events at startup and not require persistent memory. Despite this, a storage system on the device could improve the user experience by allowing the use of the application offline.

For its part, the **server** is the backbone of the application. It oversees receiving and processing client information, in addition to being the only access to the database. Because it not only stores all the information about events, but also about users, this element requires the most security. Data analysis is another of its functions, which allows us to understand what the users' needs are for the application.

Finally, the **database** manages the data saving of all events, in addition to the users' credentials. Its synchronization with the applications and their respective local storages (if any) is crucial since users should not see excessively outdated information.

4. Results

Successfully, we have design a multiplatform mobile application through Xamarin.Forms, an open source interface framework, which allows to compile applications natively for Android, iOS and Windows 10 through a single shared code [1], which in this case has been C#. In practice, it has only been tested on different Android and Windows 10 versions, but not on iOS due to lack of means.

Currently, the application allows creating and modifying events through a simple and intuitive user interface, but it does not allow sharing events with other users. Despite this, the proposal and voting date system has been satisfactorily developed and tested. Below are some screenshots of the application interface in its Android version, where you can easily see the electronic calendar elements entered.

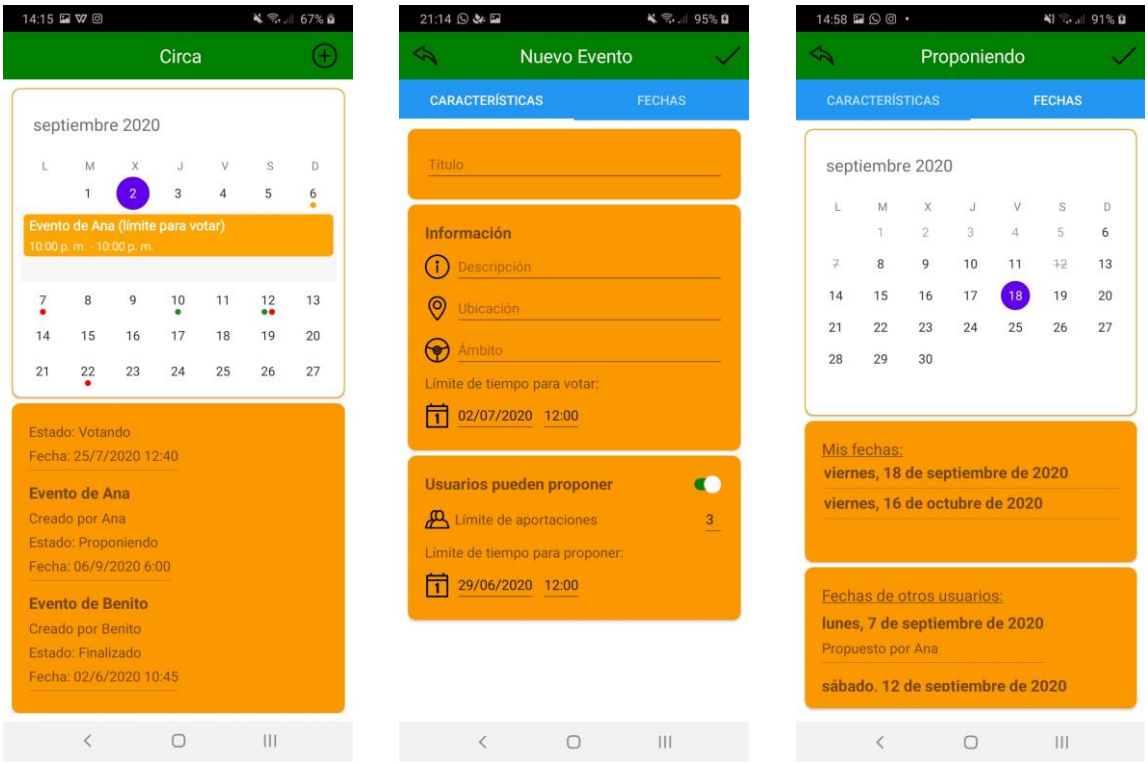

Illustration 2. Main page (left) and event management page (middle, right), in its Android version

Although no practical advance has been made on the server side, various login and cloud storage options have been explored. On this last point, a detailed model of how the Google Calendar API could be used has been developed, in such a way that all the parameters of the events and their management are transferred to the systems of the North American company.

5. Conclusions

The mobile application has effectively achieved the integration of the interface of electronic calendars with the functionality of schedulers. Its interface is intuitive and easy to use, either in its mobile (Android, iOS) or desktop (Windows 10) version. Regarding its functionality, it has been possible to effectively introduce the possibility that guests can propose their own options, instead of the traditional system, which is limited to those offered by the administrator.

Due to an undervaluation of time, certain promised capabilities have not been carried out effectively, especially on the server side. Despite this, the research carried out on cloud storage reveals that it is only possible not to depend on your own server and database, but may be recommended in certain situations. Along these lines, the integration of the system with Google Calendar services has been studied, which could be a good starting point for further work.

6. References

[1] Microsoft Docs, «What is Xamarin? | .NET », 2020, 2020. [Online]. Available at: https://dotnet.microsoft.com/learn/xamarin/what-is-xamarin. [Accessed: Jun 22, 2020].

Índice de la memoria

Índice de figuras ..... ***IV***

Índice de tablas ..... ***VII***

Índice de códigos ..... ***VIII***

Capítulo 1. Introducción ***9***

1.1 Estado del arte 9

1.1.1 Google Calendar ("Calendar") *9*

1.1.2 Microsoft Outlook (calendario integrado) *14*

1.1.3 Appgree *16*

1.1.4 Doodle *17*

1.1.5 Bash *22*

1.1.6 Tabla resumen *24*

1.2 Motivación 26

1.3 Objetivos 27

1.4 Metodología y plan de trabajo 27

1.4.1 Recursos *28*

Capítulo 2. Arquitectura del sistema ***30***

2.1 Aplicación 31

2.2 Almacenamiento local 31

2.3 Servidor 32

2.4 Base de datos 32

Capítulo 3. Funcionalidad y tecnologías ***33***

3.1 Funcionalidad básica 33

3.2 Tecnologías 35

3.2.1 Marco de desarrollo: Xamarin.Forms *35*

3.2.2 Patrón de arquitectura: MVVM *37*

3.2.3 Entorno de desarrollo: Visual Studio 2019 ..... *40*

Capítulo 4. Modelo de datos e interfaz ***43***

4.1 Modelo 43
4.1.1 Usuarios: AppUser *44*
4.1.2 Eventos *45*
4.1.3 Opciones *48*
4.1.4 Votos: UserVote *49*
4.2 Vista y Modelo de vista 50
4.2.1 Página de Aplicación: App *52*
4.2.2 Página Principal: MainPage *53*
4.2.3 Página de Evento: GenericEventPage *58*

Capítulo 5. Integración del sistema 81
5.1 Servicio dedicado 81
5.1.1 La base de datos dedicada *82*
5.2 Uso de servicios externos 85
5.2.1 Autenticación de usuarios *86*
5.2.2 Integración con Google Calendar ("Calendar") *88*

Capítulo 6. Análisis de Resultados 95
6.1 Pruebas 95
6.2 Resultados 96
6.2.1 Interfaz *96*
6.2.2 Gestión de datos *96*
6.2.3 Integración con el sistema en la nube *97*

Capítulo 7. Conclusiones 98
7.1 Objetivos no logrados 98

Capítulo 8. Trabajos Futuros 100
8.1 Posibles Mejoras 100
8.2 Ampliaciones futuras 100

Capítulo 9. Relación del proyecto con los ODS 102
9.1 Relación con el proyecto 103
9.1.1 Trabajo decente y crecimiento económico (objetivo #8) *103*
9.1.2 Industria, innovación e infraestructuras (objetivo #9) *104*

Capítulo 10. Bibliografía 105

ÍNDICE DE FIGURAS

Figura 1. Interfaz mensual de Google Calendar, versión de navegador 10
Figura 2. Detalle de la interfaz de Google Calendar, donde se muestran los calendarios ... 11
Figura 3. Ventana de creación/modificación de evento de Google Calendar 12
Figura 4. Vista general de la página web de la API de Google Calendar 14
Figura 5. Interfaz mensual del calendario de Outlook, versión de escritorio (Windows 10) 15
Figura 6. Vista general de Appgree, en su versión Android 17
Figura 7. Vista del canal Somos lo que comemos 17
Figura 8. Ventana principal de Doodle, en su versión Android 18
Figura 9. Ventana de creación de evento de Doodle, en su versión móvil 20
Figura 10. Ventana de voto de una encuesta de fecha, en su versión móvil 21
Figura 11. Precios y características de los distintos paquetes ofrecidos por Doodle 22
Figura 12. Ventana de creación de evento de Bash 23
Figura 13. Diagrama GANT del proyecto 28
Figura 14. Arquitectura del sistema 30
Figura 15. Cronograma de uso estando la opción de proponer deshabilitada 34
Figura 16. Cronograma de uso estando la opción de proponer habilitada 34
Figura 17. Comparativa entre Xamarin y Xamarin.Forms [17] 36
Figura 18. Logos de C# y .NET 37
Figura 19. Esquema de funcionamiento del patrón MVVM 38
Figura 20. Ventana integrada de Visual Studio 2019 para instalar paquetes NuGet 41
Figura 21. Diversas vistas del componente Calendar de Syncfusion Calendar [27] 42
Figura 22. Esquema del modelo 44
Figura 23. Esquema simplificado de las páginas y sus modelos de vista 51
Figura 24. Página principal de Circa 54
Figura 25. Detalle de la página principal, donde se muestra la barra de navegación 54
Figura 26. Detalle de la página principal, donde se muestra el calendario de eventos 55

Figura 27. Detalle de la página principal, donde se muestra la lista de eventos 57
Figura 28. Pestaña de características de la página de evento 59
Figura 29. Detalle de la página de evento, donde se muestra la barra de navegación..... 59
Figura 30. Detalle de la página de evento, donde se muestra la barra de desplazamiento .. 60
Figura 31. Detalle de la página de evento, donde se muestra la información general 61
Figura 32. Ejemplo de introducción de datos en un elemento Entry..... 62
Figura 33. Ejemplos de entrada de datos en elementos Picker..... 63
Figura 34. Ejemplos de entrada de datos en elementos DatePicker y TimePicker, respectivamente 64
Figura 35. Muestra de los dos estados del Switch..... 65
Figura 36. Página "Nuevo Evento", configurada para que sólo el administrador pueda proponer fechas..... 66
Figura 37. Página "Nuevo Evento", configurada para que los invitados puedan proponer fechas 67
Figura 38. Detalle de la página "Nuevo Evento", donde se muestra el calendario de fechas 68
Figura 39. Detalle de la página "Nuevo Evento", donde se muestra la lista con las fechas del usuario 69
Figura 40. Detalle de la página "Nuevo Evento", donde se muestra la lista con las fechas de los otros usuarios 70
Figura 41. Cronograma de uso estando la opción de proponer habilitada 70
Figura 42. Página "Proponiendo"..... 71
Figura 43. Detalle de la página "Proponiendo", donde se muestra el calendario de fechas 72
Figura 44. Detalle de la página "Proponiendo", donde se muestra la lista con las opciones de los otros usuarios 73
Figura 45. Cronograma de uso estando la opción de proponer deshabilitada 74
Figura 46. Cronograma de uso estando la opción de proponer habilitada 74
Figura 47. Página "Votando", configurada para que sólo el administrador pueda proponer fechas 75

Figura 48. Página “Votando”, configurada para que los invitados puedan proponer fechas 76
Figura 49. Detalle de la página “Votando”, donde se muestra el calendario de votos..... 77
Figura 50. Detalle de la página “Votando”, donde se muestra la lista con las opciones de los usuarios..... 78
Figura 51. Página “Finalizado” 80
Figura 51. Diagrama simplificado de comunicación entre usuarios con un servidor dedicado 82
Figura 52. Base de datos relacional para servidor dedicado..... 83
Figura 53. Diagrama simplificado de comunicación entre usuarios empleando sistemas externos..... 86
Figura 54. Venta de inicio de sesión de Doodle, en su versión Android 88
Figura 55. Vista mensual de Google Calendar, donde se aprecian los eventos de un calendario compartido que guarda la información de un evento de Circa 93
Figura 56. Objetivo de Desarrollo Sostenible [38]..... 102

ÍNDICE DE TABLAS

Tabla 1. Resumen de las características principales de las aplicaciones mencionadas en el Estado del arte 26
Tabla 2. Comparativa entre las tablas principales de la BD y las clases del modelo 83
Tabla 3. Comparativa entre las tablas auxiliares de la BD y los diccionarios del modelo .. 84
Tabla 4. Traspaso de las propiedades de GenericEvent a las clases de Google Calendar ... 90
Tabla 5. Traspaso de parámetros de DateEvent a la clase Event de Google Calendar 91
Tabla 6. Traspaso de propiedades de UserVote a las propiedades de la clase Attendee de Google Calendar 92
Tabla 7. Traspaso de los valores de TypeKey a los de ResponseStatus 92

ÍNDICE DE CÓDIGOS

Código 1. Fragmento de una página XAML en el que se instancia un Timepicker 39
Código 2. Fragmento del viewmodel TimePickerExampleVM 39
Código 3. Fragmento de un diccionario de recursos, en el que se instancia la clase de estilo ButtonExample 52
Código 4. Fragmento de una página XAML, en el que se muestra la instancia de un Button 52

Capítulo 1. INTRODUCCIÓN

La organización y gestión de tareas siempre han acompañado al ser humano, ya sea a través de calendarios, agendas, listas de tareas, etc. Con la llegada de la era digital, estas herramientas comenzaron a ser sustituidas paulatinamente por sus homólogos electrónicos. Las ventajas que presentan respecto a sus versiones convencionales son considerables: portabilidad, sincronización automática entre dispositivos (ordenadores de sobremesa y portátiles, tabletas inteligentes, teléfonos móviles…), posibilidad de compartir de forma sencilla entre usuarios, etc. Actualmente, estas herramientas se utilizan prácticamente en cualquier ámbito de la sociedad, tanto en lo privado como en lo público, convirtiéndose así en elementos indispensable de nuestras vidas.

Por ello, es comprensible que año tras año se creen nuevas aplicaciones móviles que pretendan aportar nuevas características y funcionalidades a las ya existentes en el mercado. En este marco se encuentra el desarrollo de este proyecto, el cual pretende crear una aplicación multiplataforma que introduzca una serie de innovaciones en el ámbito de las aplicaciones de gestión de eventos en grupo, ya sea en el apartado gráfico, funcional o estructural.

1.1 ESTADO DEL ARTE

A continuación, se enumerarán algunas de las aplicaciones que poseen características y funcionalidades similares o idénticas a las buscadas a la hora de desarrollar el proyecto. Se han seleccionado en total cinco herramientas de gestión, entre las que se incluyen calendarios electrónicos (Google Calendar y Outlook), plataformas colaborativas de trabajo (Slack) y organizadores colectivos (Doodle y Bash).

1.1.1 GOOGLE CALENDAR ("CALENDAR")

Es el nombre que recibe el servicio gratuito de agenda y calendario electrónico de la compañía norteamericana Google [2]. Sacada al mercado en 2006 en calidad de Beta, y

considerada como una aplicación terminada desde 2009, Calendar se ha convertido en uno de los grandes refrentes de aplicaciones de calendario sincronizado en línea. Entre sus puntos fuertes destacan:

1.1.1.1 Interfaz

La aplicación permite presentar los eventos de cuatro formas: diaria, semanal, mensual y anual. Esto permite al usuario, de manera rápida e intuitiva, visualizar su agenda de la manera que mejor le convenga en cada momento. Respecto a cómo se presentan los eventos en la pantalla, estos se diferencian fácilmente entre sí a través del código de colores integrado en la propia aplicación, que diferencia el calendario asociado a ese evento.

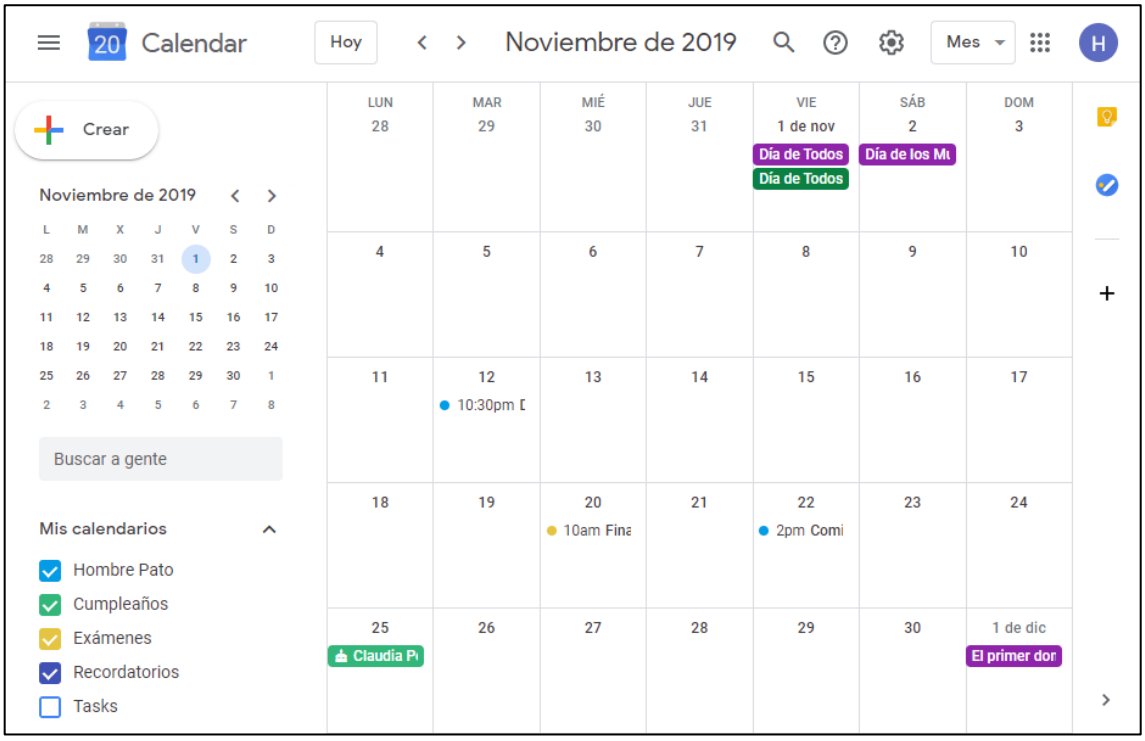

Figura 1. Interfaz mensual de Google Calendar, versión de navegador

1.1.1.2 Conjuntos de fechas

Calendar permite dividir los eventos en grupos, denominados "calendarios", los cuales se encuentran en las pestañas "Mis calendarios" y "Otros calendarios". La diferencia entre estos dos conjuntos es que el segundo engloba los calendarios compartidos —o más bien

importados, ya que no pueden modificarse— de otros usuarios, y los de intereses, que sólo pueden ser importados y son creados previamente por Google. Estos últimos incluyen los calendarios de días no lectivos, festividades religiosas y deportes, entre otros.

Por su parte, la pestaña "Mis Calendarios" incluye las agrupaciones de eventos de índole privada, como pueden ser cumpleaños de familiares, citas médicas o fechas de exámenes. Esta es una de las características que mayor valor aporta al usuario, ya que le da total libertad para crear tantos conjuntos (calendarios) como desee, además de poder compartirlos a través de la propia aplicación con otros usuarios, cambiar su código de colores o hacerlos aparecer y desaparecer de forma sencilla.

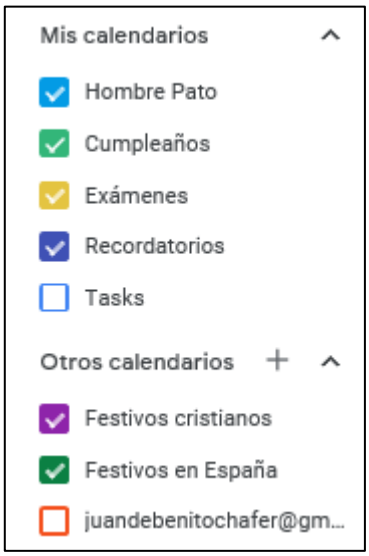

Figura 2. Detalle de la interfaz de Google Calendar, donde se muestran los calendarios

1.1.1.3 Eventos

Respecto a la personalización de los eventos, existen una gran variedad de opciones a seleccionar, entre las que se encuentran el nombre, la fecha y hora y la posibilidad de añadir notificaciones. Además de las ya nombradas, Calendar trae algunas características más novedosas, como compartir el evento a otros usuarios a través de sus cuentas de Gmail (el servicio de correo electrónico de Google), asociar el evento a un sub-calendario

determinado, permitir la visibilidad y modificación del evento a terceros y adjuntar archivos, como documentos de texto enriquecido, presentaciones o PDF.

De forma adicional, Calendar permite establecer si el evento deberá marcar al usuario como ocupado o no durante el periodo de actividad. Esto es muy práctico a la hora de organizar eventos grupales, ya que permite a conocidos, ya sean compañeros de trabajo, familiares o amigos, conocer de antemano si la persona podrá asistir o no a la actividad a la que desea invitarles. Esto no quiere decir en absoluto que la información del evento sea pública, aunque esta opción también está disponible si el usuario lo desea.

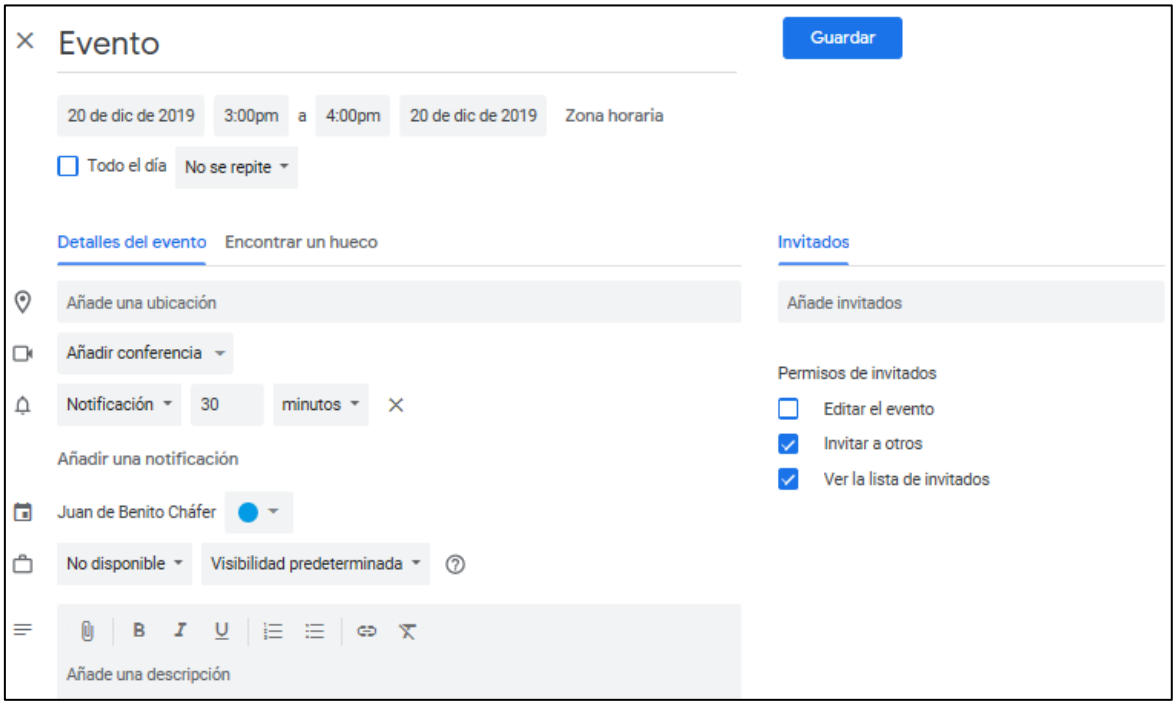

Figura 3. Ventana de creación/modificación de evento de Google Calendar

1.1.1.4 Multiplataforma

Google Calendar, siguiendo la línea general de la compañía, se puede obtener de manera gratuita en todas las plataformas relevantes; ya sea a través de sus aplicaciones nativas, disponible para móviles y tabletas Andorid, iOS o Windows 10 Mobile; o su versión web. Esta última, además de estar también adaptada para el acceso móvil, es compatible con ordenadores de escritorio y portátiles. Pese a la gran diferencia de dispositivos en los que se

encuentra disponible, Calendar presenta una interfaz prácticamente idéntica en todas las plataformas, lo que reduce el tiempo de aprendizaje necesario para manejarla con soltura.

Como es común en los servicios ofrecidos por Google, es criterio indispensable poseer una cuenta de Gmail para emplear Calendar.

1.1.1.5 Público objetivo y versiones de pago

Pese a que la mayor parte del público identifica al paquete de aplicaciones de Google como un servicio totalmente gratuito, lo cierto es que existe una versión de pago dirigida a empresas denominada G Suite [3]. Este paquete de servicios, que varía según el tamaño de la compañía, ofrece muchas funciones adicionales en un gran número de sus aplicaciones más conocidas, como Drive y Hangouts, además de incluir otros servicios completamente nuevos, como la Consola de Administración o Work Insights. Respecto a Calendar, Google únicamente ofrece un servicio adicional, que es la posibilidad de integrar las salas de reunión en el sistema.

1.1.1.6 API abierto

Uno de los mayores atractivos para los desarrolladores es la facilidad que tienen los servicios de Google para integrarse dentro de sus aplicaciones, como se puede observar en el extendido uso de Google Maps en aplicaciones de movilidad urbana, como Cabify [4] o Zity [5], entre otros. Al igual que Maps, Calendar dispone de un API (Interfaz de Programación de Aplicaciones) abierto con una gran capacidad de integración. De esta manera, la API de Google Calendar permite a aplicaciones de terceros crear calendarios, gestionar eventos e, incluso, utilizar su propia interfaz.

Google Calendar API [6] ofrece guías escritas, vídeos explicativos y un extenso índice de funcionalidades dedicado a cada uno de los lenguajes de programación soportados (Java, PHP, .NET, JavaScript, NodeJs, Ruby, Python, Go, Android, iOS) junto a ejemplos de uso. Este amplio abanico de lenguajes no obliga a los desarrolladores a implementar uno específico para utilizar las funcionalidades de Calendar, sino que es la aplicación la que se adapta a las necesidades de los programadores.

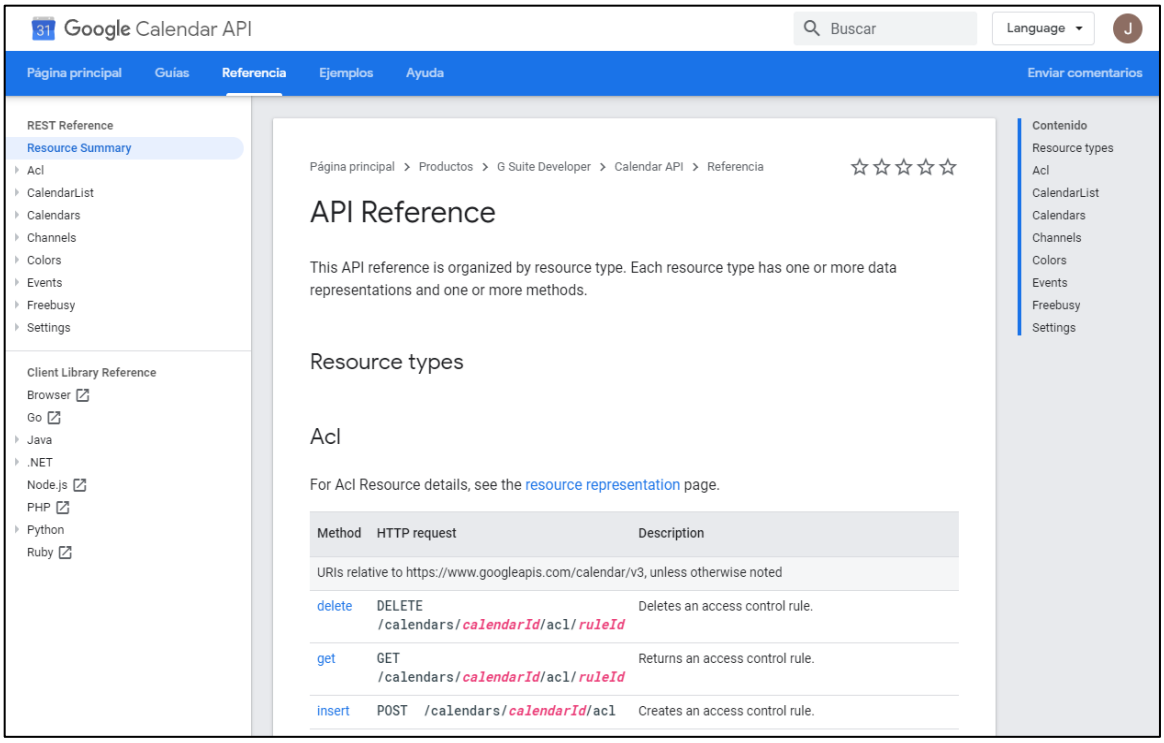

Figura 4. Vista general de la página web de la API de Google Calendar

1.1.2 MICROSOFT OUTLOOK (CALENDARIO INTEGRADO)

Es un gestor de información personal creado por Microsoft. Salió al mercado en 1997 como parte del paquete Microsoft Office, conocido actualmente como Office 365. Esta aplicación contiene, en la misma interfaz, gestor de correo electrónico, calendario, agenda y programador de tareas. Pese a que esta aplicación no es la más utilizada dentro del paquete Office, presenta una gran integración con las principales aplicaciones de la compañía, como son Word, Excel o PowerPoint, lo que la hace idónea para ser utilizada en un ámbito de tipo empresarial.

1.1.2.1 Diferencias notables entre Google Calendar y Microsoft Outlook

Debido a que ambas aplicaciones presentan prácticamente las mismas características de cara al usuario en utilidades e interfaz, el análisis de Outlook no será tan exhaustivo como el realizado para Google Calendar. Por esta razón, únicamente se centrará en las diferencias entre ambos:

- En Outlook la utilidad del calendario es accesoria a la del correo, al contrario que en Google Calendar, que es independiente. Esto posiblemente radica en el uso tan ligado que tienen el correo y el calendario en el ámbito laboral a la hora de establecer reuniones o fechas de entrega, entre otros.
- Google Calendar, para ordenadores sobremesa y portátiles, únicamente está disponible en su versión web, evitando así tener que instalar software adicional en el equipo. Por otro lado, Outlook dispone de una aplicación de escritorio, lo que evita la dependencia de una conexión activa a internet.

De forma similar a Google, Microsoft dispone de su propia API, llamada Microsoft Graph REST, la cual permite acceder a las funcionalidades de su paquete Microsoft 365 [7]. El apartado de Graph sobre Outlook permite acceder a una gran variedad de las funcionalidades del calendario integrado, lo que permite crear y leer eventos del usuario, organizar calendarios y proponer reuniones.

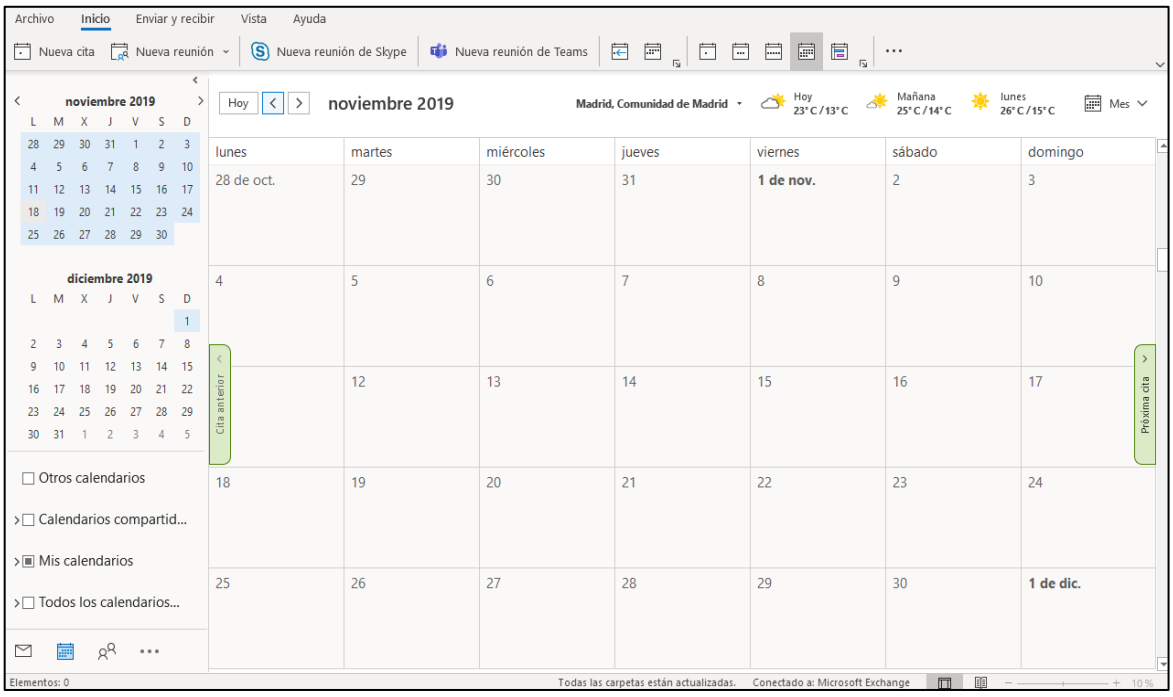

Figura 5. Interfaz mensual del calendario de Outlook, versión de escritorio (Windows 10)

1.1.3 APPGREE

Es una plataforma online española que permite canalizar propuestas de un número elevado de usuarios [8]. Hasta hace unos años se podía encontrar en versión web y como aplicación en la App Store y en Google Play. Pese a que su acceso público actualmente se encuentra prácticamente en desuso, su página web se mantiene en funcionamiento. El partido político Podemos [9], el canal más seguido de la plataforma con casi 64.000 usuarios, realizó su última aportación en 2016. Por su parte, el propio canal de la aplicación, *Descubre Appgree*, tuvo su última entrada en 2018 [10].

Su premisa es clara, pero su funcionamiento no es trivial: realizar encuestas rápidas para conocer la opinión de un gran conjunto de usuarios (del orden de miles o millones incluso) de una forma rápida e imparcial [10]. Esto lo consigue a través de un algoritmo estadístico propio, apodado *DemoRank*, que obtiene el conjunto de respuestas más apoyadas por los usuarios.

Como se ha mencionado antes, la plataforma está prácticamente en desuso, posiblemente por la privacidad de sus algoritmos, que muchos usuarios pudieran haber tomado como una falta de transparencia por parte de los administradores de los canales. Pese a ello, y como anuncian es su página web oficial [10], su servicio orientado a empresas, Appgree 4Biz, se utiliza en la actualidad dando servicio a grandes compañías, como el banco Santander, Indra o Orange. Respecto a sus precios, utilidades y ámbito de trabajo, la página web es totalmente opaca.

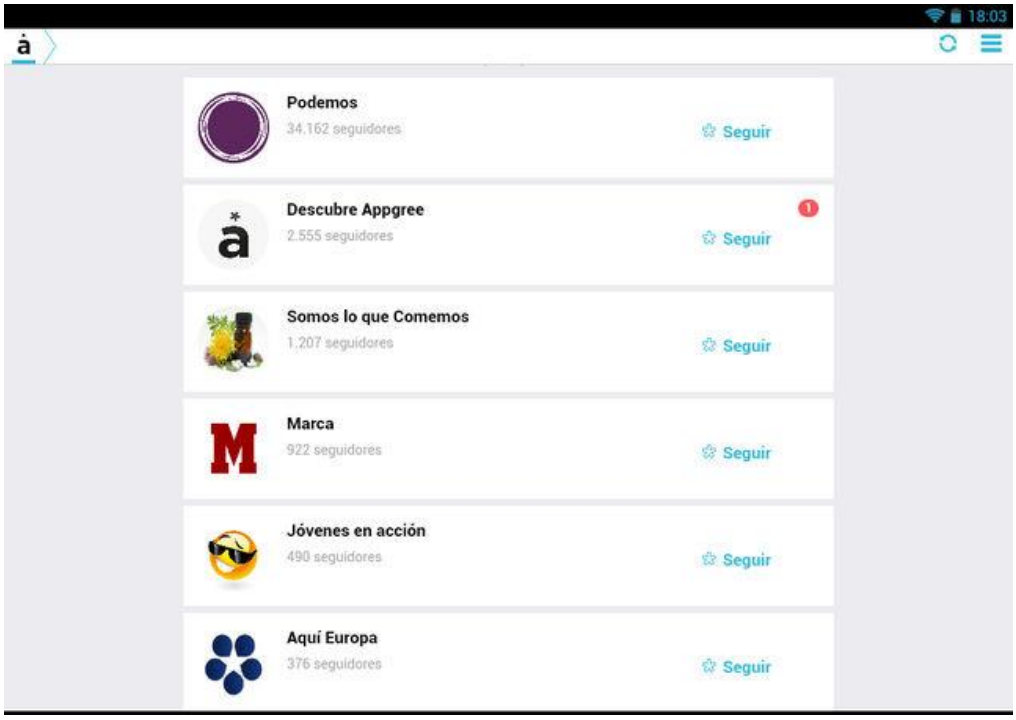

Figura 6. Vista general de Appgree, en su versión Android

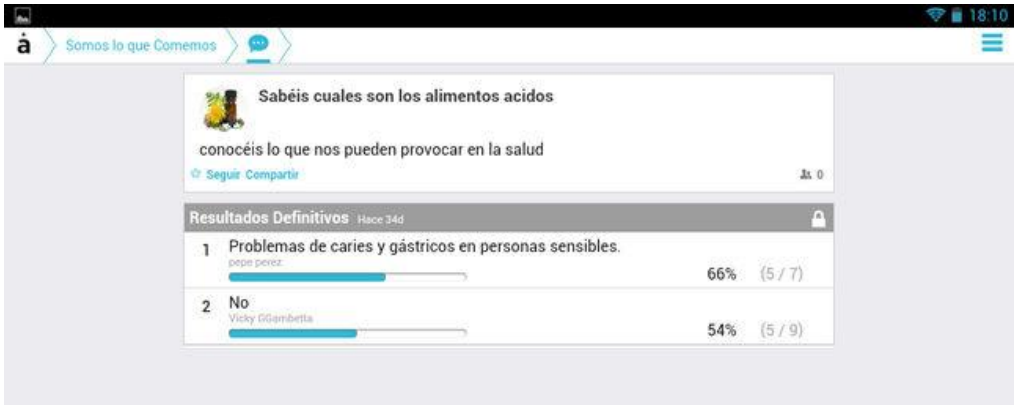

Figura 7. Vista del canal Somos lo que comemos

1.1.4 DOODLE

Se trata de una herramienta de gestión de tiempo (conocidas como *schedulers* en inglés) pensada para organizar encuentros grupales. Desde el año 2014 forma parte de la multinacional suiza Tamedia tras adquirir el 100% de sus acciones [11].

1.1.4.1 Interfaz

Doodle posee una interfaz sencilla, aunque con funcionalidades limitadas. Esto permite a los usuarios familiarizarse rápidamente con la aplicación, que mantiene este formato simplificado en sus versiones móviles (iOS y Android) y de navegador. La principal desventaja que presenta es que no existe ninguna ventana en la que se muestren las posibles fechas de todos los eventos que se están votando, lo que puede confundir al usuario al no dejarle la opción de comparar las opciones que mejor le convengan con la comodidad debida.

Figura 8. Ventana principal de Doodle, en su versión Android

1.1.4.2 Encuestas

La aplicación permite a un usuario (administrador) crear dos tipos de encuestas: de fecha, en las cuales la índole del evento es seleccionada previamente por el administrador y los usuarios invitados seleccionan sus preferencias, y las de texto, en las cuales el administrador

hace una pregunta y el resto de los usuarios invitados responden entre un grupo de opciones establecidas por él. Ambas encuestas permiten una gran variedad de opciones, como seleccionar la ubicación, escribir notas sobre el evento, establecer un límite de votos o mantener el voto oculto, entre otros.

Tras el proceso de creación, el administrador invita a los usuarios a la votación, ya sea a través de correo electrónico, enlace directo o servicios de mensajería instantánea, como WhatsApp y Facebook Messenger. Una gran opción que posee la aplicación es que no es estrictamente necesario tener una cuenta Doodle para votar en un evento, lo que la hace muy llamativa para un uso casual. De manera adicional, cada evento permite utilizar un chat que invita a los usuarios a discutir sobre el evento, haciéndose así innecesaria la comunicación fuera de la aplicación.

Pese a todas estas características, Doodle no permite que sean los propios usuarios invitados los que contribuyan en establecer las opciones de la encuesta, ya que no permite que estos seleccionen fechas u opiniones (en el caso de las encuestas de texto) no establecidas por el administrador previamente.

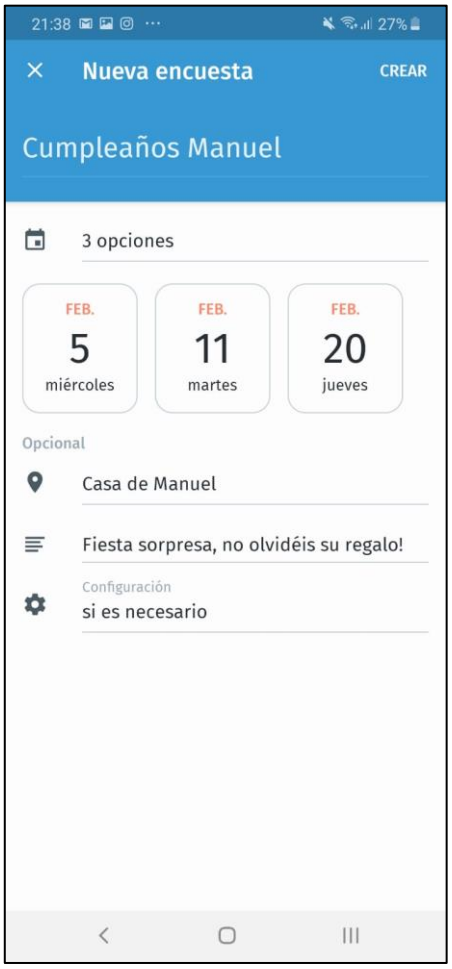

Figura 9. Ventana de creación de evento de Doodle, en su versión móvil

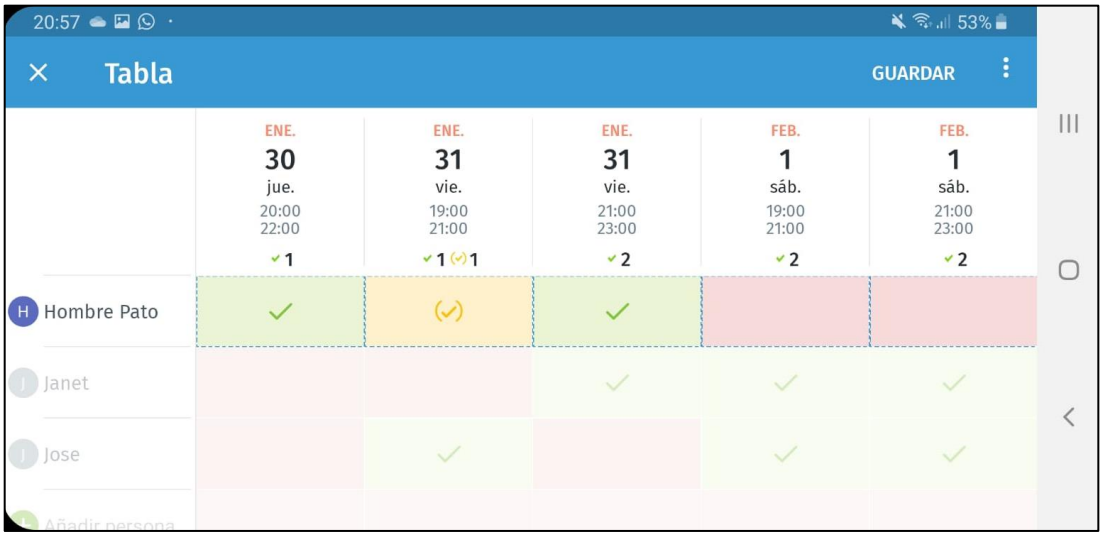

Figura 10. Ventana de voto de una encuesta de fecha, en su versión móvil

1.1.4.3 Multiplataforma

Pese a que originalmente la aplicación únicamente se encontraba disponible en navegadores, desde 2014 Doodle se puede descargar para dispositivos Android e iOS [12].

1.1.4.4 Público objetivo y versiones de pago

Actualmente, Doodle posee una versión gratuita y cuatro de pago [13]. Por un lado, la gratuita está pensada para ser utilizada de forma casual e incluye las características anteriormente mencionadas. Por el otro, las versiones de pago, entre las que se encuentran la *Starter*, *Pro*, *Team* y *Enterprise*, incluyen una serie de mejoras progresivas. Pese a que la primera (*Starter*) está orienta al uso personal, pero intensivo, de la aplicación, las tres últimas (*Pro*, *Team* y *Enterprise*) lo están al uso empresarial.

Entre las opciones más interesantes destacan la posibilidad de sincronizar automáticamente las fechas de las encuestas con calendarios digitales, como los de Google Calendar o Outlook; la posibilidad de recibir notificaciones, y la sincronización automática de cambios en la encuesta —con la versión estándar, cuando el administrador realiza algún cambio en la encuesta, los usuarios que ya han votado no reciben ninguna notificación—.

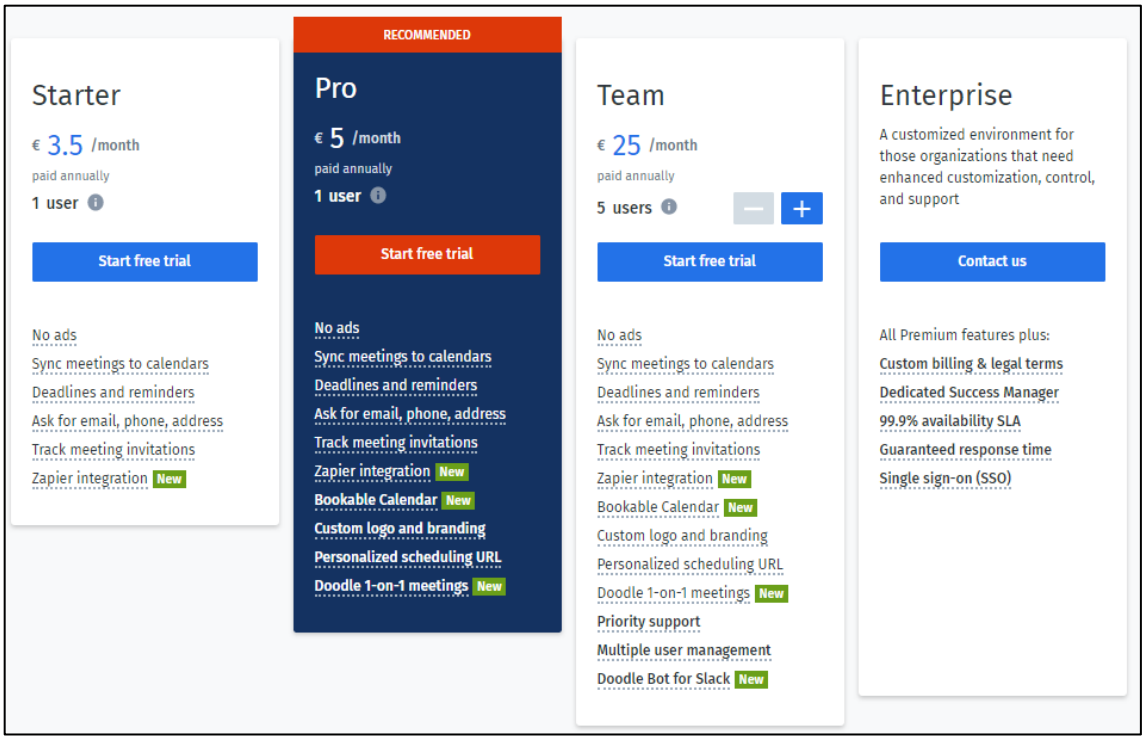

Figura 11. Precios y características de los distintos paquetes ofrecidos por Doodle

1.1.5 BASH

Se trata de un *scheduler* móvil para organizar "quedadas informales" con amigos y familiares [14]. Publicada en mayo de 2019, fue creada por un grupo internacional de emprendedores europeos (Filz et al.). Pese a que en la actualidad posee menos de 1000 usuarios, Bash muestra funciones interesantes en el campo de los organizadores de eventos.

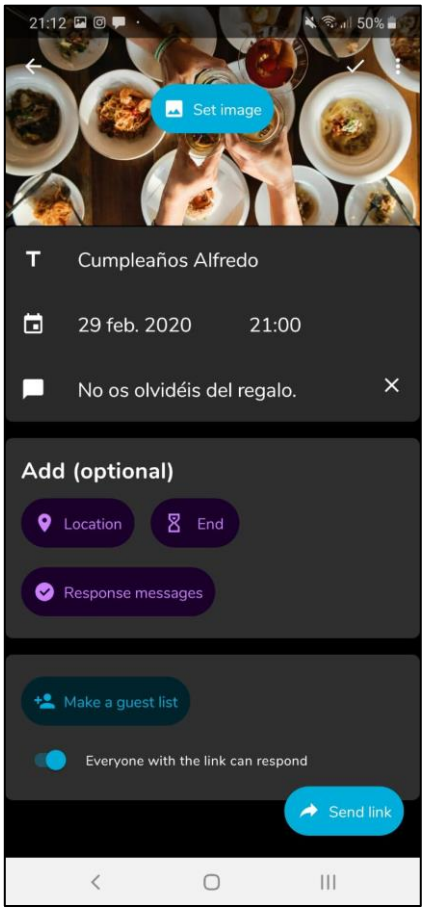

Figura 12. Ventana de creación de evento de Bash

1.1.5.1 Diferencias notables entre Doodle y Bash

Debido a las similitudes entre ambas aplicaciones, únicamente se mencionarán las características más novedosas y las diferencias más importantes entre Doodle y Bash:

- La interfaz de Bash se apoya mucho en elementos visuales dinámicos, en lugar de los tradicionales campos estáticos que emplea Doodle.
- La funcionalidad es más limitada, ya que el administrador no puede seleccionar varias fechas posibles, limitándose a una exclusivamente. Por ende, la única función de los invitados es marcar su asistencia al evento en fecha propuesta. Esto no es una desventaja necesariamente, ya que, debido a ello, se reducen las complicaciones relacionadas con la gestión del evento, lo que podría resultar en una mejor experiencia de usuario.
- Bash da un lugar principal al chat de grupo, que se encuentra en la pantalla principal del evento y que se posiciona como el elemento central de *feedback* entre los usuarios

y el administrados del evento. Por su parte, Doodle "esconde" su chat en una ventana separada.

- La versión de navegador de Bash no posee una funcionalidad completa, ya que se limita a permitir a los invitados votar y escribir mensajes en el chat sin necesidad de tener una cuenta. Esto quiere decir, que, para gestionar y administrar el evento, la necesidad de la aplicación móvil es imperativa.
- Bash da un lugar principal al chat de grupo, que se encuentra en la pantalla principal del evento y que se posiciona como el elemento central de *feedback* entre los usuarios y el administrados del evento. Por su parte, Doodle "esconde" su chat en una ventana separada.
- Al contrario que la versión gratuita de Doodle, Bash avisa de manera predeterminada de los cambios de estado del evento (fecha, nuevos mensajes…) en cuento se producen. Esto lo puede hacer a través de la aplicación (que requiere una cuenta de Google o Outlook) o directamente a través del correo electrónico.
- Bash no posee versiones de pago, lo que tiene sentido si se piensa en que parece estar orientada a encuentros informales entre amigos y familiares.
- Bash no posee una versión en español, lo cual es relativamente común en una aplicación recién salida al mercado. Esto podría ser una barrera de entrada a un gran número de clientes potenciales.
- Según indican algunos de los comentarios de los usuarios en la página oficial de la Play Store, la aplicación tiene algunos bugs y *glitches* [14]. Por sus respuestas, el equipo de desarrollo parece bastante dirigido a solucionar todos los problemas emergentes.

1.1.6 TABLA RESUMEN

La siguiente tabla indica, de una forma visual e intuitiva, las características principales de las aplicaciones previamente descritas.

	Google Calendar	*Outlook Calendar*	*Appgree*	*Doodle*	*Bash*
Interfaz de calendario mensual	X	X			
Interfaz multitemporal (día, semana, mes…)	X	X			

Creación de (sub)calendarios	X	X		X	
Eventos privados	X	X		X	X
Eventos compartidos “de dictador”	X	X		X	X
Eventos compartidos con participación			X	/	/
Multiplataforma	X	X	/	X	/
Archivos adjuntos	X	X			
Mostrar estado (ocupado, libre…)	X	X		X	X
Orientado al uso personal	X	X		X	X
Orientado al uso empresarial	X	X		X	
Red social			X		
Recordatorio automático de eventos	X	X		$	X
Estructurado en comunidades			X		
Versiones *premiun*	X	X	X	X	

Versión independiente	X			X	X
Compartir (sub)calendarios de eventos	X	X		$	
Chat integrado				X	X
API público	X	X			

Tabla 1. Resumen de las características principales de las aplicaciones mencionadas en el Estado del arte

Respecto a los símbolos empleados, su significado es que la aplicación:

- "X": Posee la característica
- "/": Posee la característica, pero con una funcionalidad parcial
- "$": Posee la característica, pero únicamente en una versión de pago

1.2 MOTIVACIÓN

Pese a que existen numerosas aplicaciones de organización de tareas (conocidas como *schedulers* en inglés), como las ya mencionadas Doodle o Bash, su falta de funcionalidad en algunos aspectos o su alta complejidad de utilización pueden hacer que muchos usuarios sientan que no existe en el mercado una aplicación que se adapte a sus necesidades de la forma correcta. Por esta razón, este proyecto pretende aportar nuevas utilidades, características y perspectivas que permitan al usuario utilizar la aplicación de una manera más natural e intuitiva.

Con este propósito, la aplicación a desarrollar presentará las siguientes características y funcionalidades novedosas respecto a los *schedulers* actuales:

- La comúnmente utilizada lista de eventos, que es la interfaz estándar de este tipo de aplicaciones, será sustituida por un calendario electrónico, similar al de Google Calendar o Outlook. En él aparecerán las posibles fechas de todos los eventos del segmento temporal seleccionado. De esta manera el usuario pueda visualizar de una

manera más natural todos sus eventos, lo que le permitirá no tener que seleccionar cada evento por separado para ver las fechas disponibles de cada uno.

- Los usuarios invitados del evento tendrán la posibilidad (si así lo habilita el administrador), de proponer sus propias opciones al evento al que estén inscritos. De este modo el administrador puede llamar a la colaboración, haciendo el proceso de votación mucho más agradable, ya que, en el caso de no encontrar su opción en el listado, podrían incorporarla sin dificultades.
- El creador del evento podrá nombrar a otros usuarios como administradores, concediéndoles derechos adicionales, como modificar los atributos del evento o invitar a nuevos usuarios. La utilidad efectiva de esta característica está por determinar y es posible que finalmente no se desarrolle, aunque se tendrá en cuenta.

Con estas características, pretendemos llegar a un público casual, alejado del ámbito laboral, ya sean nuevos en este tipo de aplicaciones o ya hayan empleado herramientas similares, como las ya mencionadas Doodle o Bash.

1.3 OBJETIVOS

A continuación, se presentan los objetivos del proyecto:

- Realizar un análisis comparativo y objetivo de las funcionalidades y características de las aplicaciones anteriormente descritas, dejando abierta la posibilidad de incluir en el análisis otras no mencionadas. Además de estas, se estudiarán las siguientes características:
  - Los usuarios invitados tendrán la posibilidad de proponer opciones adicionales al evento a las ya impuestas por el administrador.
  - La integración de una interfaz de tipo calendario electrónico, en lugar de la ampliamente utilizada lista de eventos.
  - Los creadores del evento tendrán la opción de seleccionar a otros usuarios como administradores. De esta manera, existirá la opción de separar los usuarios en dos grupos con roles diferentes.
- Conseguir un diseño tanto conceptual como gráfico del sistema que permita habilitar el conjunto de características deseadas, como permitir a los usuarios proponer opciones al evento, de tal forma que el sistema pueda ser escalable tanto en funcionalidades como en número de usuarios.
- Implementar de forma práctica una parte demostrativa del diseño anterior con el objetivo de que un administrador pueda compartir eventos con una serie de usuarios.

1.4 METODOLOGÍA Y PLAN DE TRABAJO

El plan de trabajo del proyecto se dividirá en nueve partes:

1) **Estado del arte**: Se realizará un análisis comparativo de las tecnologías actuales con el objetivo de acercarse al mercado de las aplicaciones móviles de calendario electrónico y organizador de eventos en grupo.
2) **Selección de tecnologías**: Se estudiarán los API, programas multiplataforma, lenguajes de programación y demás recursos técnicos para poder seleccionar los más provechosos a fin de lograr los objetivos propuestos en la sección anterior.
3) **Diseño conceptual**: Se creará un primer esbozo del funcionamiento del sistema.
4) **Diseño de arquitectura del sistema**: Se fijarán las ideas del diseño conceptual en diagramas de flujos de trabajo (*flowchart*).
5) **Diseño gráfico**: Se realizará un diseño preliminar del apartado visual (interfaz).
6) **Programación del diseño gráfico**: Se programará la interfaz no funcional del sistema en el lenguaje seleccionado previamente.
7) **Integración con el sistema elegido**: Se procederá a integrar la interfaz con las funciones de la/las API seleccionadas.
8) **Validación de la integración almacenamiento en la nube del sistema**: Se probará el correcto funcionamiento del sistema, ya con la base de datos plenamente funcional.
9) **Redacción de la memoria del proyecto**: Se finalizará con la redacción de la memoria del proyecto, que incluirá todos los apartados previamente mencionados.

De acuerdo con los puntos descritos anteriormente, se ha realizado un diagrama GANT que organice temporalmente los periodos de trabajo correspondientes.

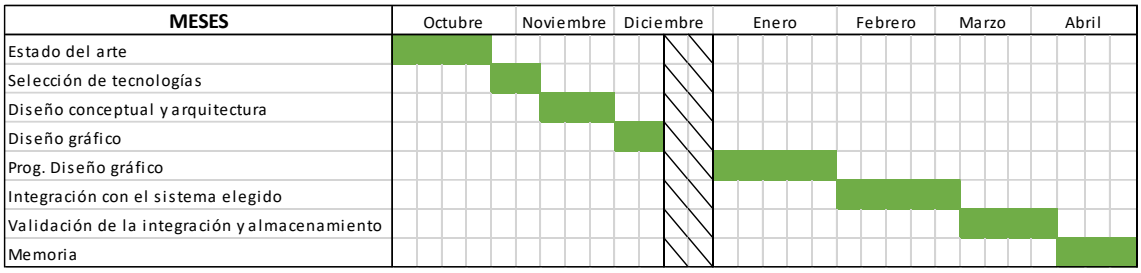

Figura 13. Diagrama GANT del proyecto

1.4.1 RECURSOS

Para la consecución de los objetivos anteriormente propuestos se hará uso de las siguientes herramientas.

1.4.1.1 Elementos de hardware

El desarrollo principal del proyecto se basará en dos elementos principales: un ordenador para diseñar y programar la aplicación, y un teléfono móvil inteligente para probarla. Debido

a los medios disponibles, el **ordenador** del que se hará uso será d**e sobremesa (sin marca comercial)** y dispondrá de un sistema operativo Windows 10 Pro. Por su parte, el teléfono inteligente será un **Samsung Galaxy A50**, el cual cuenta con un sistema operativo Android 9.0. También se realizarán pruebas con el simulador de Android Studio, en el cual se ha instalado un Píxel 2 con la misma versión de Android que el terminal físico.

De forma adicional, se emplearán:

- **Ordenar portatil HP Elitebook 810 G3**: Se utilizará como ordenador de apoyo fuera del ambiente normal de trabajo y como ordenador principal si se viera la necesidad. Debido a que los documentos de la aplicación serán compartidos en línea, la transición de trabajo entre ambos no debería traer complicaciones innecesarias.
- **Teléfono móvil inteligente Samsung S6**: El uso de este dispositivo, que cuenta son un sistema operativo Android 7.0, será muy reducido. Su función en el desarrollo del proyecto será comprobar el buen funcionamiento de la aplicación en sistemas Android más anticuados.

1.4.1.2 Elementos de software

Los programas y tecnologías por emplear se determinarán y describirán en el apartado de Tecnologías.

Capítulo 2. ARQUITECTURA DEL SISTEMA

En este capítulo se discutirá la estructura general de la arquitectura del sistema, donde se explicarán los distintos elementos que lo integran, además de realizar una breve descripción de estos. Tras un riguroso estudio, se ha concluido que el sistema buscado es el representado en la figura siguiente.

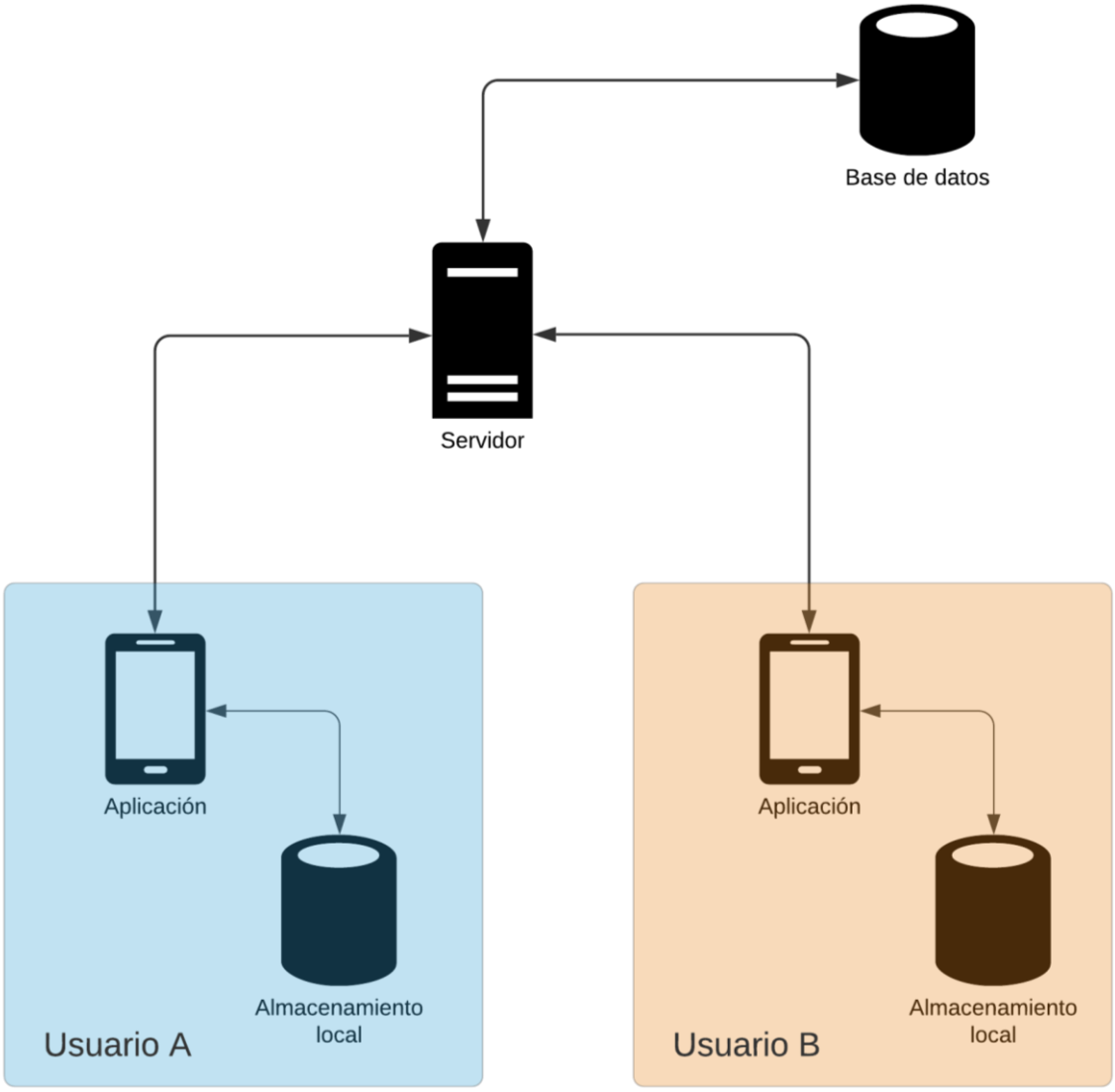

Figura 14. Arquitectura del sistema

El fin de este sistema, y por tanto del proyecto, es permitir que un administrador (*admin*) pueda crear un evento y que pueda compartirlo con otros usuarios para que visualicen sus características, voten determinadas opciones o, incluso, propongan las suyas propias. Para mantener la información de los eventos actualizada en todo momento es necesario implementar un sistema similar al de la figura, formado por los elementos descritos a continuación.

2.1 APLICACIÓN

La aplicación realiza la función de ser la interfaz de usuario, es decir, supone la vía de acceso de los usuarios a los eventos y a toda la funcionalidad relacionada con los mismos. También es la encargada de recibir todos los cambios sobre los eventos que el usuario tenga compartidos, asegurándose de que la información de la base de datos y el almacenamiento local está actualizada en todo momento. Para su diseño se empleará una interfaz intuitiva y sencilla, similar a las empleadas en otros *schedulers* como Doodle o Bash, pero con ciertas características de los calendarios inteligentes.

2.2 ALMACENAMIENTO LOCAL

El almacenamiento local es una extensión de la aplicación que guarda de forma persistente en el dispositivo (local) la información de los eventos del usuario. La idea inicial es que se trate de una base de datos relacional idéntica en estructura (o con mínimas modificaciones) a la localizada en el servidor. Las dos bases de datos deberían estar sincronizadas de forma periódica para que la información local esté lo suficientemente actualizada. La otra opción sería no utilizar ningún almacenamiento local, lo que requeriría pedir la información de los eventos al servidor cada vez que arrancara la aplicación, de forma similar a una aplicación web. Pese a que no permite el uso de la aplicación offline y podría suponer una sobrecarga en el servidor, este diseño evitaría problemas sobre la integridad de los datos, al asegurar que la aplicación se encuentra siempre actualizada.

Pese a lo mencionado anteriormente, es necesario tener un grado mínimo de persistencia a la aplicación, principalmente para guardar las preferencias y credenciales del usuario. Estas

últimas son esenciales, ya que, al guardarse, evitan al cliente tener que introducir su usuario y contraseña cada vez que accede al sistema. Para este tipo de parámetros no se suelen emplear bases de datos, ya que son campos de datos de muy poco tamaño.

2.3 SERVIDOR

El servidor, como su nombre indica, es el encargado de recibir y procesar las peticiones de los clientes (la aplicación). Es el punto de unión entre la aplicación y la base de datos, lo que lo convierte en el eje central de la arquitectura del sistema. No sólo es el encargado de autentificar y almacenar las credenciales de los usuarios, sino también de gestionar la entrada y salida de información de la base de datos. La seguridad del servidor es un tema crucial, ya que, si se burlara, la base de datos podría quedar expuesta o inutilizada. Además, deberá de estar preparado para recibir cientos o miles de peticiones por minuto, dependiendo del número de usuarios y de los eventos que se hayan creado hasta el momento.

El control de acceso a la base de datos no es la única función del servidor. También es muy importante el análisis de datos, ya que revela el comportamiento de los usuarios frente a la aplicación. De esta forma, se podría utilizar la información obtenida para mejorar la experiencia de usuario, ya sea modificando la interfaz o introduciendo sugerencias personalizadas, entre otras funciones.

2.4 BASE DE DATOS

La base de datos central guarda la información de todos los usuarios y eventos del sistema. Su sincronización con las aplicaciones y sus respectivos almacenamientos locales (si los hubiera) es crucial, ya que los usuarios no deberían ver información excesivamente desactualizada.

Capítulo 3. FUNCIONALIDAD Y TECNOLOGÍAS

En este capítulo se explicará la funcionalidad básica de la aplicación móvil, para luego poder justificar los lenguajes de programación, el diseño gráfico y la arquitectura seleccionados para su desarrollo. De ahora en adelante, a la aplicación presentada en este proyecto se la denominará "Circa", lo que facilitará su diferenciación respecto a otros servicios y aplicaciones referenciados en esta memoria.

3.1 FUNCIONALIDAD BÁSICA

Al igual que la mayor parte de *schedulers*, el objetivo primordial de Circa es que un administrador (*admin*) pueda crear un evento y que tenga la posibilidad de poder compartirlo con otros usuarios para que puedan apreciar sus características y realizar cambios en las mismas. Su función principal, además de mostrar información general del mismo —como un título o una descripción corta—, es poder presentar una serie de fechas posibles en las cuales puede o no tener lugar, para que, posteriormente, los usuarios invitados seleccionen las que mejor les convengan. El objetivo de esto es que se decida, de una forma democrática, la fecha o conjunto de fechas que más usuarios respalden. Para lograrlo, es necesaria la introducción de dos fechas fundamentales que marcarán el ritmo del evento:

- **El límite de voto**: Marca el periodo máximo del que disponen los usuarios invitados (y el propio administrador) para seleccionar las fechas que mejor les convengan de las permitidas. Es la fecha fundamental del evento.
- **El límite de propuesta**: Indica el periodo máximo en el cual los invitados pueden proponer las fechas deseadas en las cuales les gustaría que el evento tuviera lugar. Pese a que abre la puerta a la funcionalidad más novedosa de la aplicación respecto a las de su clase, su uso es opcional.

Pese a que la definición de ambos pueda resultar similar, lo cierto es que son radicalmente diferentes. En el caso de que únicamente existiera la primera —que es la estrictamente necesaria—, el administrador poseería todo el poder a la hora de decir el conjunto de fechas

que los invitados tienen para votar. Si bien es cierto que esto no deja sin voz a los usuarios, reduce en gran medida su campo de actuación. De esta manera, podría producirse una situación en la que una fecha le es conveniente a la mayoría, pero no se encuentra en el listado propuesto y no tienen la "oportunidad" de presentarla. Esto es lo común entre los *schedulers* del mercado, como Doodle.

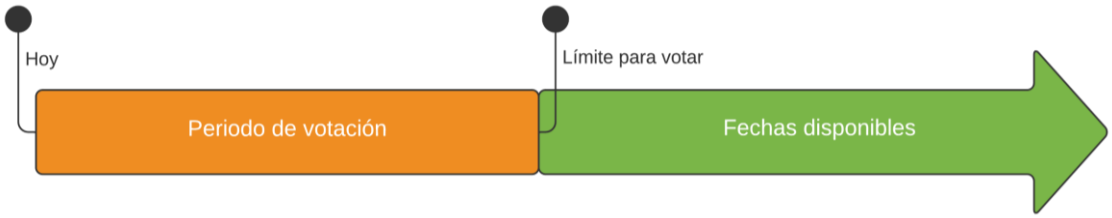

Figura 15. Cronograma de uso estando la opción de proponer deshabilitada

Al añadir la fecha de propuesta eliminamos la limitación impuesta a los usuarios, invitándoles a que propongan las fechas que ellos mismos desearían, para luego iniciar el sistema de votación convencional. Si bien está claro que sus fechas no tienen por qué ser las mejor valoradas —las propuestas por el administrador podrían ser las más convenientes en muchos casos—, al menos tienen conciencia de que han podido votar las que ellos deseaban con total libertad. De forma adicional, este sistema se complementa con un número de fechas máximo a proponer para cada usuario, el cual es establecido por el administrador —al que también afecta—.

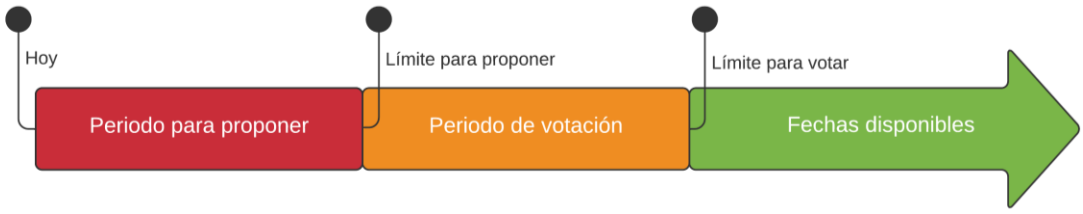

Figura 16. Cronograma de uso estando la opción de proponer habilitada

Además de las capacidades anteriormente descritas, el evento contiene otros parámetros opcionales existentes en otros gestores de agenda, los cuales son:

- **Título**: Ayuda a identificar fácilmente el evento en el listado.

- **Descripción**: Aporta detalles adicionales sobre el evento.
- **Ubicación**: Indica el lugar donde el evento va a tener lugar. Puede dejarse en blanco.
- **Ámbito**: Señala el tipo de evento creado. Permite agrupar los eventos en conjuntos fácilmente diferenciables, como "Amigos" o "Familia", entre otros.

Estos campos son directamente gestionados por el administrador, y no existe la posibilidad de dar a los invitados la posibilidad de cambiarlos. Poseen una peculiaridad, y es que pueden ser modificables en cualquier periodo de vida del evento. Por su parte, el límite de voto y propuesta únicamente son modificables al crear el evento. La razón de ello se explicará en el apartado sobre la clase *DateOption*.

3.2 TECNOLOGÍAS

En este apartado se discutirán todas las tecnologías empleadas en el desarrollo de Circa.

3.2.1 MARCO DE DESARROLLO: XAMARIN.FORMS

Xamarin.Forms es un marco de desarrollo multiplataforma que permite desarrollar aplicaciones en Xamarin.Android, Xamarin.iOS y UWP (Plataforma Universal de Windows en español) a través de un único código compartido. Lanzada al mercado en mayo de 2014 [16] por Xamarin, se trata de una extensión de la plataforma de desarrollo .NET (*dotnet* en inglés) de Microsoft dirigida al entorno de los teléfonos inteligentes. Su característica más atractiva es que permite programar un único código en C# (C-Sharp) que se compila en cada sistema operativo de forma nativa, lo que reduce los tiempos de carga y reduce el uso de memoria, aumentando la productividad [1].

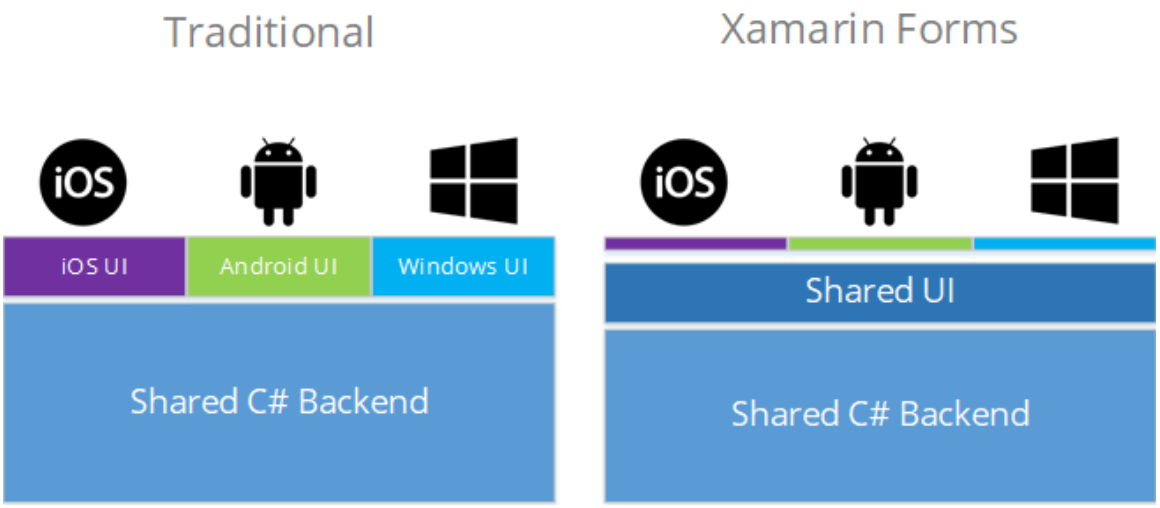

Figura 17. Comparativa entre Xamarin y Xamarin.Forms [17]

No se deben confundir Xamarin y Xamarin.Forms. Pese a que el primero supuso un avance a la hora de poder mantener un único código compartido C# en el *Backend*, todavía existía la necesidad de desarrollar la interfaz para cada sistema operativo por separado en sus lenguajes nativos. Con Forms esta necesidad desaparece, ya que, al compilar el proyecto, la interfaz programada en XAML y C# se transforma automáticamente a los lenguajes nativos correspondientes. Cabe mencionar que, para ciertas funcionalidades específicas, y en especial si se tratan de control de hardware, la utilización de código nativo es ineludible.

Esta plataforma encaja con las necesidades de nuestro proyecto, ya que permite emplear un único conjunto de lenguajes —en este caso, XAML y C-Sharp— para los dos principales sistemas operativos móviles en la actualidad: Android e iOS. Adicionalmente, Forms también es compatible con la Plataforma Universal Windows (UWP), lo que aumenta el alcance del proyecto a todos los dispositivos con Windows 10. Debido a que no se han requerido accesos especiales a hardware u otros servicios específicos, las tasas de código compartido deberían ser cercanas al 100%, aunque no se descartan posibles modificaciones futuras que afecten a ello.

3.2.1.1 XAML

XAML es un lenguaje de marcado basado en XML creado para sostener la interfaz de la WPF (Base de Presentación de Windows) [18]. Su principal diferencia respecto a HTML es

que sus definiciones representan directamente la creación de tipos de instancias de objetos en su clase parcial, que normalmente funciona con código C#. Es decir, al declarar cualquier elemento, como un botón (*Button* en XAML), un objeto C# *Button* es creado en el código de su clase parcial. La relación entre ambas clases parciales es tan fuerte que existe la opción de diseñar toda la interfaz en una única clase C#, aunque no es recomendable por la complejidad que entrañaría.

3.2.1.2 C-Sharp (C#)

C# es un lenguaje de programación multiparadigma fuertemente tipificado. Está basado, como cabría esperar por su nombre, en C, y presenta una gran similitud con muchos de los lenguajes de programación orientado a objetos más populares, como C++, Java y JavaScript [19]. Fue creado por Windows para establecerlo como lenguaje de *backend* en la anteriormente mencionada plataforma .NET. La idea detrás de C# fue la de crear un código de alto nivel capaz de funcionar en cualquier tipo de dispositivo, ya sean móviles, tabletas, ordenadores, smartwatch o consolas de videojuegos, entre otros. Es uno de los lenguajes soportados por la UWP, cuyo objetivo es crear aplicaciones universales que puedan funcionar en cualquier dispositivo con Windows 10 [20].

Figura 18. Logos de C# y Microsoft .NET [21]

3.2.2 PATRÓN DE ARQUITECTURA: MVVM

El patrón de arquitectura seleccionado ha sido el modelo-vista-modelo de vista (*model–view–viewmodel* en inglés), comúnmente conocido como patrón MVVM. Presentado por

John Gossman en 2005, se trata de una variación del conocido patrón modelo-vista controlador —conocido por sus siglas en inglés MVC— ajustado a la WPF [22]. El uso de este patrón en Circa no ha sido casual, ya que es el recomendado por Microsoft para crear aplicaciones en Xamarin [1], al igual que en muchas de sus otras plataformas de desarrollo.

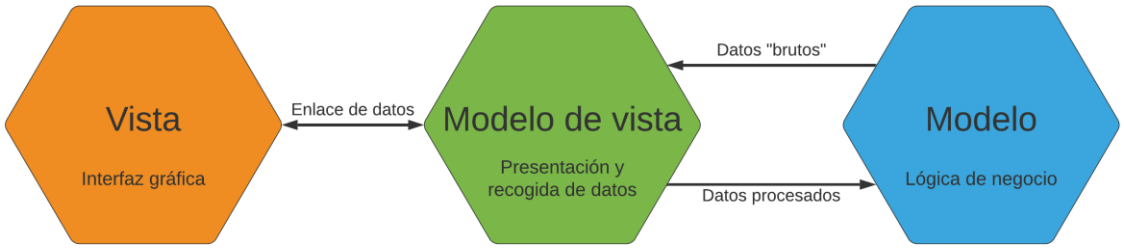

Figura 19. Esquema de funcionamiento del patrón MVVM

De igual forma que el patrón MVC, el objetivo del MVVM es desacoplar en la medida de lo posible la interfaz de usuario (*view*) de la lógica de la aplicación (*model*) [23]. Así, el modelo de vista (*viewmodel*) actúa entre ambos como un filtro, ya sea preparando los objetos para su visualización o recogiendo los parámetros introducidos por el usuario. Por su parte, la vista únicamente debe preocuparse por mantener la interfaz y algunos eventos menores que se enfoquen exclusivamente en el apartado visual, alejándose lo máximo posible del trato directo con los objetos del modelo.

3.2.2.1 Data Binding

Xamarin soporta el llamado *Data Binding* (o enlace de datos en español), que es una comunicación directa que se establece entre los elementos del *frontend* —como el *Button* ya mencionado— y el *backend*. Este mecanismo no debe confundirse con la relación entre las dos clases parciales tratada en el apartado sobre XAML, ya que el *Data Binding* está orientado a relacionar los atributos de los elementos del *view* en XAML con objetos C# del *viewmodel*. Este enlace permite mantener la parte visual de la aplicación y la lógica de clases sincronizadas de tres formas [23]:

- ***One-Way***: Los cambios en los objetos producen cambios en el elemento *frontend* asociado, pero no al revés. Es el valor por defecto en la mayor parte de los elementos XAML.
- ***OneWayToSource***: Los cambios en el elemento *frontend* producen modifican los objetos del *backend*, pero no al contrario. Esta modalidad es la menos utilizada.
- ***TwoWay***: El enlace de datos se produce de manera bidireccional. Mantiene una sincronización perfecta entre el objeto y el elemento de la interfaz, pero consume muchos recursos.

```
<Page.BindingContext>
        <TimePickerExampleVM/>
</Page.BindingContext>

<TimePicker
       x:Name="TimePickerExample"
       Time="{Binding TimePickerExampleTime, Mode=TwoWay}"
       IsEnabled= TimePickerIsEnabled, Mode=OneWay/>
```

Código 1. Fragmento de una página XAML en el que se instancia un Timepicker

```
public class TimePickerExampleVM
{
       private TimeSpan _timePickerExampleTime;
       private bool _timePickerExampleIsEnabled;

       public void ChangeShownTime(TimeSpan time)
       {
              TimePickerExampleTime = time;
       }

       public string TimePickerExampleTime
            {
                 get { return _timePickerExampleTime; }
                 set
                 {
                      _timePickerExampleTime = value;
                      RaisePropertyChanged();
                 }
            }
}
```

Código 2. Fragmento del viewmodel TimePickerExampleVM

Los dos fragmentos muestran, respectivamente, un elemento XAML *TimePicker* con sus correspondientes atributos y el *viewmodel* asociado. *TimePicker* es un elemento nativo de Xamarin.Forms cuya función principal es recoger la fecha introducida por el usuario, además

de permitir otras funciones adicionales. Se han declarado dos de sus propiedades de forma explícita: *Time* e *IsEnabled*. La primera, cuya función es recoger la fecha introducida por el usuario, se encuentra enlazada de forma bidireccional con el objeto *TimePickerExampleTime*, lo que permite que los cambios realizados por el usuario se guarden en el objeto, al mismo tiempo que deja abierta la posibilidad de que la función *ChangeShownTime()* del *viewmodel* modifique su valor. Por su parte, *IsEnabled* regula la posibilidad de que el usuario pueda o no modificar la fecha introducida. En este caso, el modo *OneWay* únicamente permite que su valor sea modificado cambiado el valor de *TimePickerExampleIsEnabled* en el *viewmodel*.

En la mayor parte de los casos, la declaración explicita del tipo de enlace no es necesaria, como es el caso de *IsEnabled*, ya que *OneWay* es su modo de enlace predeterminado. Existen casos, como el de la propiedad *Time*, en el que cambiar el modo predefinido es conveniente. Por ejemplo, si no existiera *ChangeShownTime()*, sería recomendable que el enlace de datos de *Time* fuera cambiado de su modo predeterminado *TwoWay* a *OneWay* para ahorra recursos.

3.2.3 ENTORNO DE DESARROLLO: VISUAL STUDIO 2019

Microsoft Visual Studio es un entorno de desarrollo integrado (*IDE*, por sus siglas en inglés) para sistemas operativos Windows, macOS y Linux. Su gran variedad de lenguajes de programación y de herramientas soportados lo convierten en una elección muy atractiva a los programadores, especialmente si pretenden desarrollar con lenguajes propios de Microsoft, como C# o XAML. La versión actual, Visual Studio 2019, posee una serie de herramientas "mínimas" ampliables mediante NuGet, la herramienta de gestión de paquetes de .NET [24].

Hoy por hoy, Visual Studio es el único IDE que soporta Xamarin.Forms, por lo que se trata de la única alternativa viable para el desarrollo de este proyecto. Para crear aplicaciones en Forms es condición indispensable la instalación de dos paquetes (o *NuGets*): "Xamarin.Essentials" [25] y "Xamarin.Forms" [26].

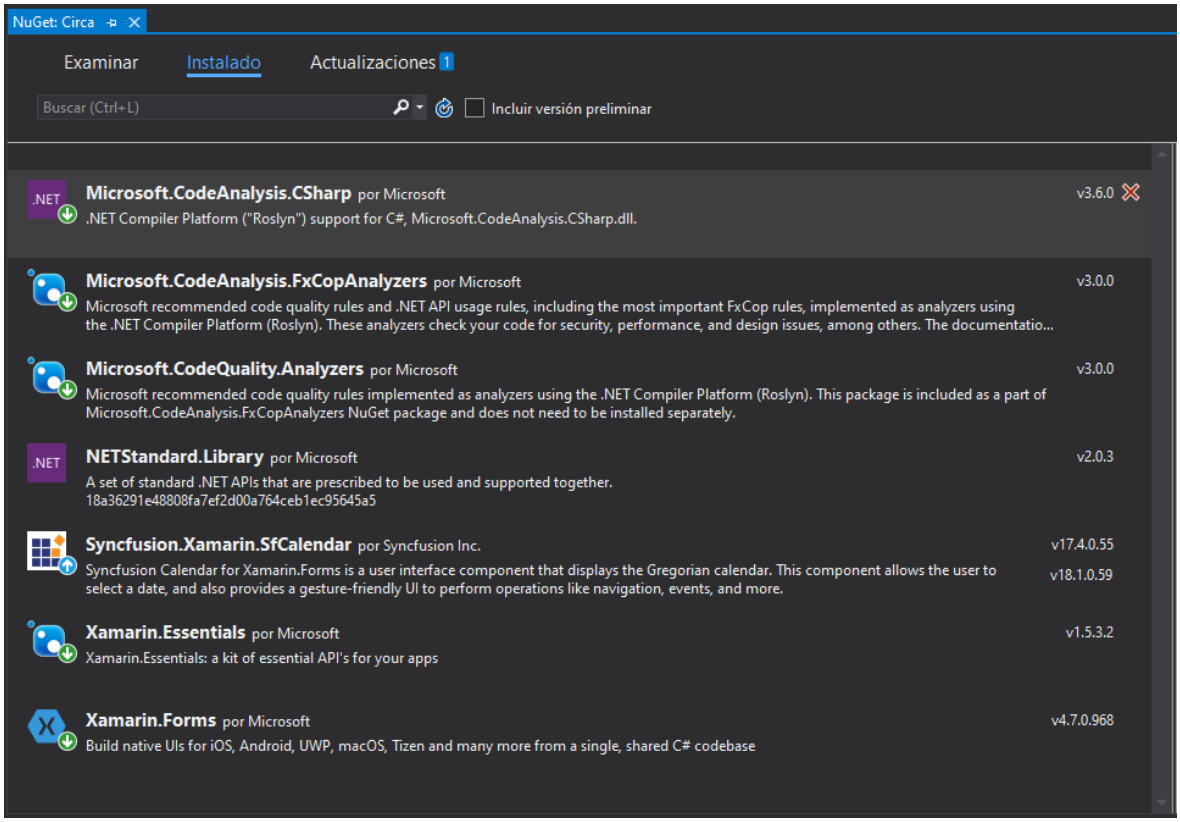

Figura 20. Ventana integrada de Visual Studio 2019 para instalar paquetes NuGet

3.2.3.1 Paquetes de terceros: Syncfusion Calendar

Ya en las primeras etapas de desarrollo, se decidió que uno de los elementos centrales de la aplicación debía ser un calendario interactivo de interfaz mensual, por lo que se comenzaron a buscar y estudiar posibles controladores de terceros, ya fueran públicos o privados. El seleccionado fue el Syncfusion Calendar, un complemento multiplataforma cuya versión para Xamarin.Forms encajaba con la funcionalidad buscada. Su uso requiere la obtención de una licencia por parte de la empresa Syncfusion, fácilmente adquirible de forma gratuita, siempre y cuando se cumplan los requisitos mencionados en su página web [27].

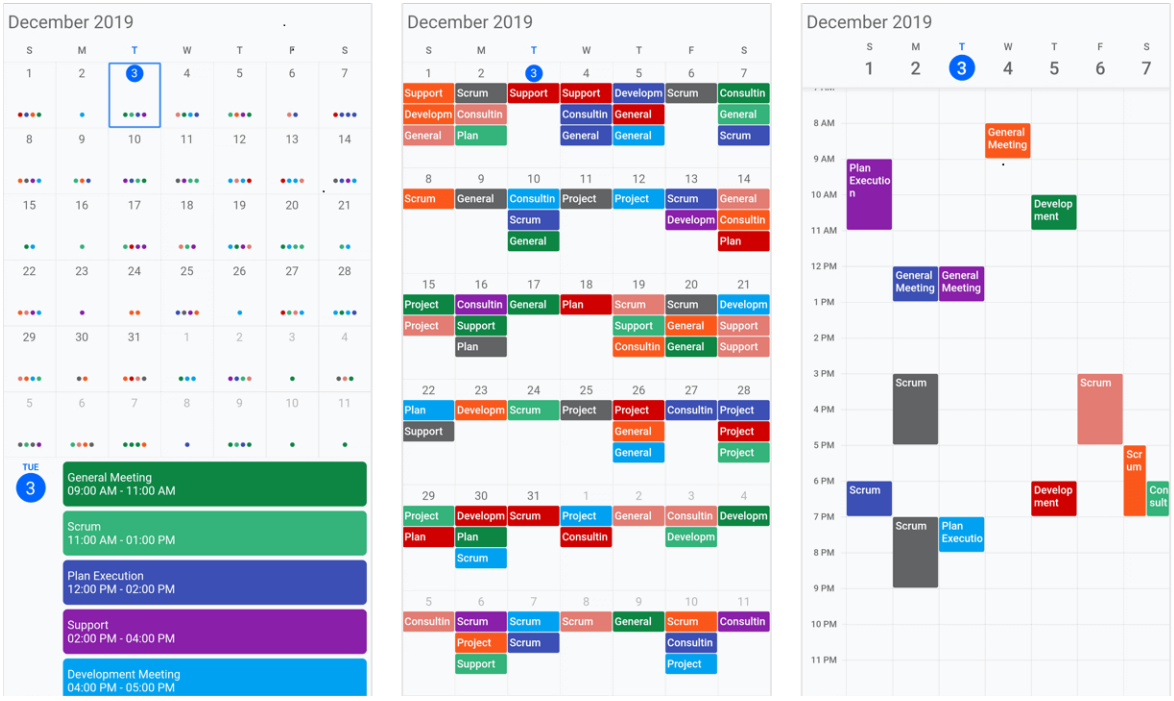

Figura 21. Diversas vistas del componente Calendar de Syncfusion Calendar [28]

Para ser añadido al proyecto se debe instalar el paquete "Syncfusion.Xamarin.SfCalendar", el cual permite el uso de un componente de interfaz de calendario gregoriano y da acceso a una amplia funcionalidad, como la navegación y acceso a gestión de eventos, entre otros [29]. Su documentación y ejemplos pueden ser encontrados en su API oficial, Syncfusion Flutter Widgets [28].

Capítulo 4. MODELO DE DATOS E INTERFAZ

Para explicar el diseño funcional y estético de Circa, se pensó en dividir este capítulo según las tres partes del patrón MVVM previamente explicado, el cual está formado por el modelo, la vista y el modelo de vista. El problema es que la fina separación entre las vistas y sus respectivos modelos es, en muchos casos, demasiado tenue para poder ser explicadas correctamente por separado. Por esta razón, se ha preferido presentarlas ambas en conjunto.

De esta forma, este capítulo tendrá dos apartados. El primero tratará íntegramente del modelo, mientras que el segundo lo hará sobre la vista y el modelo de vista en conjunto, centrándose en la interfaz y su funcionalidad.

4.1 MODELO

La capa de modelo (*model*) del patrón MVVM se corresponde con la lógica de negocio, que es la parte de la aplicación que se encarga de determinar cómo se crea, gestiona y almacena la información de un programa. Al crear el modelo de Circa, se hizo mucho hincapié en la escalabilidad, lo que explica que en el esquema inferior se muestren clases padre con una sola hija.

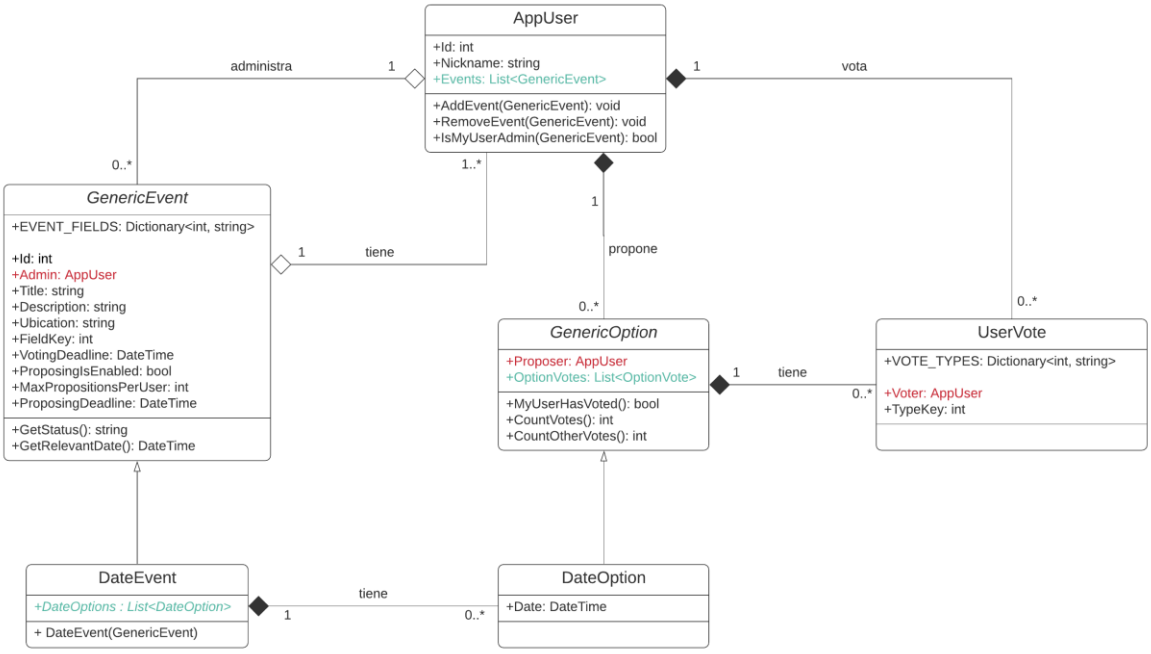

Figura 22. Esquema del modelo

El modelo de Circa se encuentra formado por seis clases, dos de ellas abstractas. Debido a ello, se ha preferido explicar el contenido de éstas en cuatro grupos: usuarios, eventos, opciones y votos.

4.1.1 USUARIOS: APPUSER

La clase *AppUser* almacena y gestiona los datos de los usuarios de la aplicación. Debido a que no se han implementado todavía las comunicaciones con el servidor, y, por lo tanto, se desconocen todavía los campos necesarios para su correcta integración en el sistema, esta clase únicamente cumple con una funcionalidad mínima. Es muy posible que se deban efectuar grandes cambios en esta clase, ya sea introduciendo nuevas propiedades y funciones, o modificando las ya existentes.

4.1.1.1 Propiedades

- **Id**: Se trata de un identificador numérico de tipo *integer*, único para cada usuario. Es un atributo "secreto" de cara al usuario, ya que queda reservado para tareas de gestión y almacenamiento. No se puede modificar.

- **Nickname**: Es una cadena de caracteres que el usuario se asigna a sí mismo para que el resto de los participantes de un evento le identifiquen. Varios usuarios podrían poseer el mismo.
- **Events**: Es la lista de eventos en los que el usuario está involucrado. Pese a que pueda parecer contraintuitivo, el evento no tiene constancia de que este usuario está adscrito a él.

4.1.1.2 Funciones

- **AddEvent(Generic Event)**: Añade un evento a la lista *Events*.
- **RemoveEvent(GenericEvent)**: Elimina un evento a la lista *Events*.
- **IsMyUserAdmin(GenericEvent)**: Devuelve un booleano que indica si el usuario en cuestión es o no el administrador del evento pasado como parámetro.

4.1.2 EVENTOS

Los eventos son el pilar fundamental de Circa, por lo que sus clases correspondientes son las más complejas y completas. Por temas de escalabilidad, este grupo está formado por una clase padre, *GenericEvent*, y una clase hija, *DateEvent*. Se trata de clases con muchas propiedades con un alto grado de personalización.

4.1.2.1 GenericEvent

GenericEvent almacena los atributos, propiedades y funciones comunes a todos los tipos de evento. Abre la puerta a la utilización futura de eventos que no discutan sólo fechas, sino otro tipo de opciones.

4.1.2.1.1 Atributos estáticos

- **EVENT_FIELDS**: Representa una colección de claves y valores irrepetibles, enlazando un código numérico de tipo *integer* (clave) con una cadena de caracteres de tipo *string* (valor). Debido a que existe un listado finito de ámbitos (“fields” en inglés) seleccionables, su uso reduce la repetición innecesaria de cadenas. En la actualidad, el diccionario clave-valor contiene las siguientes relaciones:

- [-1] = "Otro"
- [0] = "Familia"
- [1] = "Trabajo"
- [2] = "Amigos"
- [3] = "Salud"

4.1.2.1.2 Propiedades

- **Id**: Al igual que el del usuario, se trata de un *integer* que identifica el evento de forma unívoca. Se crea de forma automática y no puede cambiarse.
- **Admin**: Es una instancia de la clase *AppUser* que indica quién es el creador y administrador del evento. Pese a que bastaría con almacenar su id, se ha preferido utilizar la clase para poder acceder a su apodo de forma rápida. Por su parte, la lista de eventos es eliminada por cuestiones de memoria y privacidad.
- **Title**: Se trata de una cadena de caracteres pensada para que los participantes del evento puedan relacionar fácilmente el objetivo de este. Puede dejarse en blanco, aunque no es recomendable.
- **Description**: Es un elemento de refuerzo al título, el cual almacena información adicional.
- **Ubication:** Similar a la descripción, pero ideado para escribir la dirección o lugar donde el evento va a tener lugar.
- **FieldKey**: Empleado de forma conjunta con el diccionario *EVENT_FIELDS*, esta propiedad indica con un *integer* el ámbito del evento.
- **VotingDeadline**: Almacena la fecha y hora límite para votar, la cual marca el final del periodo de votación. Es del tipo *DateTime*, que es la clase más habitual para guardar las fechas y horas
- **ProposingIsEnabled**: Es un booleano que indica si el evento permite o no a los invitados proponer sus propias opciones.
- **MaxPropositionsPerUser**: Indica el número máximo de opciones que puede proponer cada invitado.

- **ProposingDeadline**: Almacena la fecha y hora límite para proponer opciones, la cual marca el final del periodo de proposición.

Cabe mencionar que las dos últimas propiedades únicamente son de utilidad si *ProposingIsEnabled* tiene como valor "true". Si no es el caso, su valor es el predefinido de cada propiedad. Se podrían haber creado una clase hija que incluyera exclusivamente los eventos que permiten proponer opciones a los usuarios, pero hubiera requerido una mayor complejidad en el resto del diseño, además de impedir de cierta forma la escalabilidad del mismo.

4.1.2.1.3 Funciones

- **GetStatus()**: Devuelve el momento de vida del evento, que puede ser "Proponiendo", "Votando" y "Finalizado". Existe un cuarto estado, "Nuevo Evento", que la función no puede devolver debido a que para llamarla es necesario que la clase esté creada. Este problema se soluciona en el *viewmodel*, el cual sí es capaz de discriminar si se está creando un nuevo evento o si se está modificando uno antiguo.
- **GetRelevantDate()**: En línea *GetStatus()*, el objetivo de esta función es devolver la fecha de mayor relevancia para los usuarios respecto al momento de vida actual. Por ejemplo, si el periodo de proposición ha finalizado, la fecha límite para proponer opciones ya no tiene ninguna utilidad, por lo que la función devolverá la fecha de votación.

4.1.2.2 DateEvent

Se trata de una extensión de la clase *GenericEvent*, a la que se añade una lista de opciones de fecha, clase específica de Circa que se explicará más adelante.

4.1.2.2.1 Propiedades

- **DateOptions**: Es la única propiedad nueva que añade *DateEvent* a la de su padre, pero juega un papel fundamental. Se trata de una lista de elementos del tipo *DateOption*, los cuales almacenan fechas y votos.

4.1.2.2.2 Constructores especiales

- **DateEvent(GenericEvent)**: Genera un evento de fecha a partir de un evento genérico. La lista de opciones se crea de forma predeterminada sin valores.

4.1.3 OPCIONES

Las opciones son las clases donde se almacenan la opción a votar, ya sea de fecha o de otro tipo, además de ser las encargadas de gestionar el guardado y contabilización de sus votos. De forma análoga a los eventos, este grupo está formado por una clase padre, *GenericOption*, y una clase hija, *DateOption*.

4.1.3.1 GenericOption

GenericOption almacena las propiedades y funciones comunes a todos los tipos de opción a votar. Al igual que *GenericEvent*, abre la puerta a la utilización futura de opciones que no incluyan exclusivamente fechas.

4.1.3.1.1 Propiedades

- **Proposer**: Similar a la propiedad *Admin* de *GenericEvent*, indica el usuario que ha creado la opción. En el caso de que el evento no permita proponer opciones a los invitados, el usuario siempre será el administrador.
- **OptionVotes**: Se trata de la lista de votos del evento (clase *UserVote*), en la que cada elemento indica quién ha votado y cuál ha sido su tipo de voto.

4.1.3.2 Funciones

- **MyUserHasVoted()**: Devuelve un booleano que indica si el usuario en cuestión ha votado o no. El tipo de voto no influye, simplemente revisa si el usuario está en la lista.
- **CountVotes()**: Devuelve el número de participantes que ha votado esta opción, sin importar su tipo de voto.

- **CountOtherVotes()**: Idéntico a *CountVotes()*, pero eliminando el voto de nuestro usuario —si es que éste ya había votado esa opción—.

4.1.3.3 DateOption

Se trata de una extensión de la clase *GenericOption*, a la que se añade el tipo de opción que maneja la clase padre, que en este caso es una fecha. Pese a que no hay nada que impida en su configuración guardar una hora junto a la fecha, la interfaz actual no lo permite. Esto se ha hecho para:

- Simplificar la recogida de datos y no complicar mucho la interfaz.
- Evitar problemas con horas coincidentes en el mismo día, ya sea al seleccionar la opción o para votarla.

4.1.3.3.1 Propiedades

- **Date**: Almacena el valor de la fecha y hora que se va a votar. Actualmente, esta propiedad sólo guarda días.

4.1.4 VOTOS: USERVOTE

UserVote es una clase sencilla que recoge el tipo y el usuario emisor del voto de una opción. Pese a que se podría considerar que esta clase es innecesaria, ya que podría guardarse el usuario y el voto en un diccionario clave-valor (*Dictionary<TKey,TValue>*), lo cierto es que una clase separada ofrece mayores capacidades a la hora de introducir nuevas funcionalidades. Si en el futuro se quisiera añadir un nuevo atributo al voto, podría ser muy complicado implementarla sin una clase separada. Además, los diccionarios con claves de tipos extraños —como sería en nuestro caso *AppUser*— no presentan buenos rendimientos.

4.1.4.1 Atributos estáticos

- **VOTE_OPTIONS**: Similar al diccionario *EVENT_FIELDS* de *GenericEvent*, este atributo representa una colección de claves y valores irrepetibles, enlazando un código numérico de tipo *integer* (clave) con una cadena de caracteres de tipo *string*

(valor). Con su implementación, se reducen las repeticiones innecesarias de cadenas de caracteres, asignando un número a la opción seleccionada en su lugar. Actualmente, los conjuntos clave-valor son:

- [000] = "NON_VALID_VOTING_CODE"
- [100] = "Me viene bien"
- [200] = "No me viene bien, pero podría asistir"
- [300] = "No me viene bien"

4.1.4.2 Propiedades

- **Voter**: Análogo de *Admin* de *GenericEvent y Proposer* de *GenericOption*, esta propiedad indica el usuario que ha votado la opción correspondiente.
- **OptionKey**: Empleado de forma conjunta con VOTE_OPTIONS, sirve para indicar la intención del voto del usuario, ya que el modelo de Circa permite no sólo votar, sino mostrar la conformidad o disconformidad con la opción propuesta. Actualmente, esta opción no está en uso, y todos los votos realizados se asignan a la clave “100”.

4.2 VISTA Y MODELO DE VISTA

En este apartado discutiremos el esquema de clases realizado para las clases de la vista (*view*) y su enlace con el modelo a través de sus correspondientes modelos de vista (*viewmoedel)*. Las primeras se identifican por llevar el sufijo “-Page”, y son las únicas de tipo parcial de todo el proyecto, exceptuando *GenericEventPage*, que está programada íntegramente en C# por su naturaleza abstracta. Como se puede apreciar en el diagrama inferior, todas las *pages* no abstractas tienen una relación de *Data Binding* con un *viewmodel* homónimo y el sufijo “-VM”, lo que ayuda a identificarlas. Debido a su estrecha relación es imposible explicar correctamente el funcionamiento de una vista sin su modelo, por lo que se ha decidido comentarlas de forma conjunta.

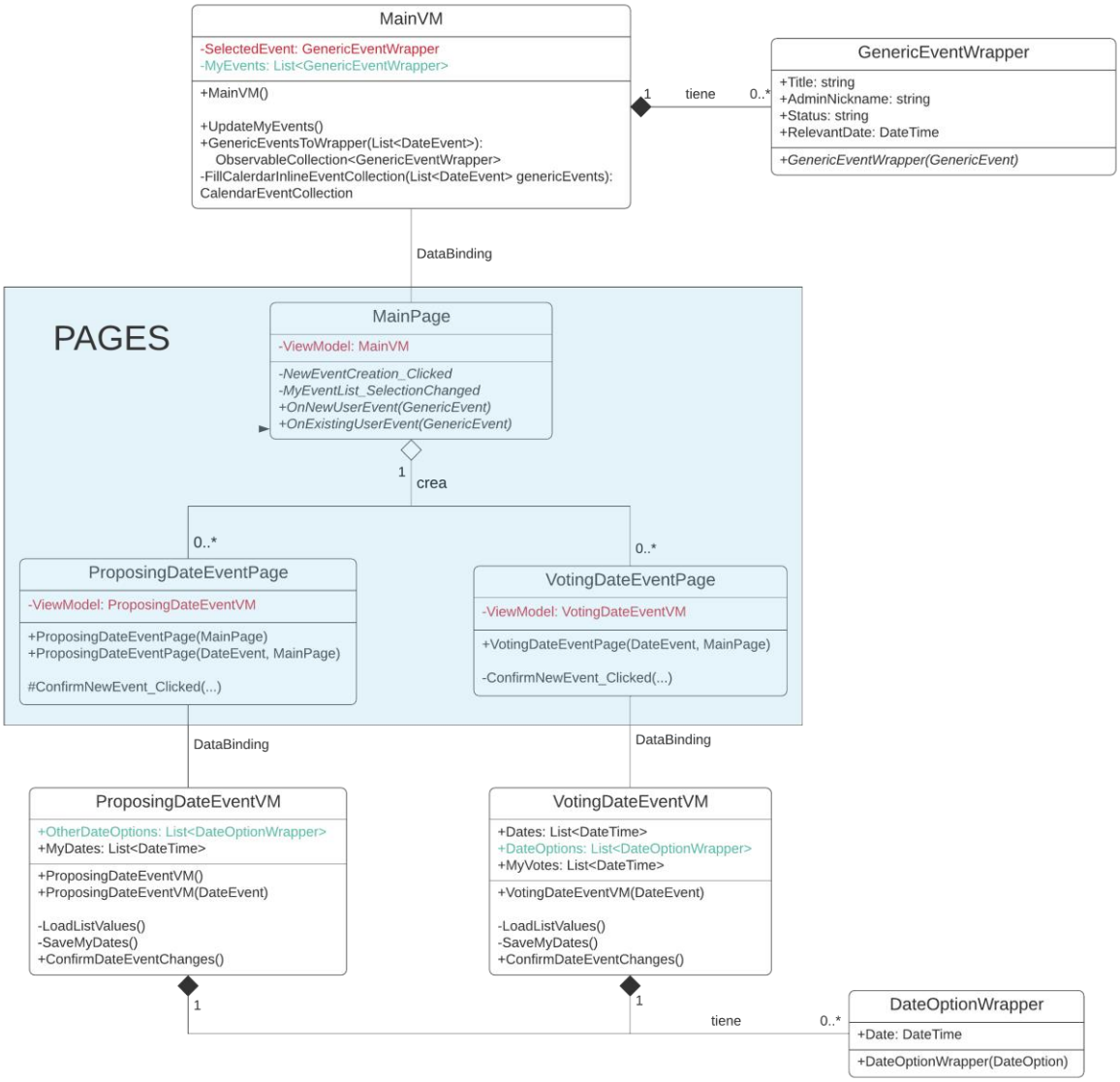

Figura 23. Esquema simplificado de las páginas y sus modelos de vista

Cabe señalar que todas las capturas mostradas de aquí en adelante se han obtenido de un Samsung A50 con sistema operativo Android 9. Por su parte, los iconos empleados han sido obtenidos de la página web de Flaticon [30], y su autoría le corresponde a Freepik. Como indica la página, el autor ha sido referenciado convenientemente en los créditos de la aplicación, respetándose así los acuerdos para el uso gratuito del paquete de iconos.

4.2.1 PÁGINA DE APLICACIÓN: APP

App es la única clase parcial que requiere ser instanciada de forma obligatoria, además de ser inmutable —siempre existe, no puede sobrescribirse en memoria—. Cabe mencionar que esta clase no está pensada para ser la central de la aplicación, sino más bien un punto de partida para inicializar los componentes y recursos compartidos de toda la aplicación.

4.2.1.1 App.xaml

Pese a que resulta contraintuitivo, no está permitido declarar elementos visuales, sino para enlazar uno o varios diccionarios con todas las páginas del proyecto. Un diccionario de recursos —o *Resource Dictionary*— es una página XAML dedicada exclusivamente a la personalización visual de los elementos de la interfaz. Al ser instanciado en el archivo *App.xaml*, el diccionario se convierte automáticamente en un recurso accesible desde cualquier otro archivo del proyecto, lo cual es muy conveniente para evitar repetir código de forma innecesaria y mantener una única estética a lo largo de toda la aplicación. Comparándolo con su homólogo en web, los diccionarios son comparables a los archivos CSS.

```
<Style x:Key="ButtonExample" TargetType="Button">
      <Setter Property="BackgroundColor" Value="Blue"/>
      <Setter Property="FontSize" Value="14"/>
</Style>
```

Código 3. Fragmento de un diccionario de recursos, en el que se instancia la clase de estilo ButtonExample

```
<Button Style=”ButtonExample”
      FontSize=”12”
      Margin=”10, 10”/>
```

Código 4. Fragmento de una página XAML, en el que se muestra la instancia de un Button

Por un lado, en el código Código 3 se ha declarado una clase de estilo de un botón (*Button*) llamando “ButtonExample” en un diccionario de recursos instanciado en *App.xaml*. De ahora en adelante, cualquier botón que se instancie con el valor “ButtonExample” en la propiedad Style poseerá las propiedades concretas de color y tamaño de letra descritas en el diccionario. Por el otro, en el Código 4 se muestra un caso particular, en el cual, además de asignar una nueva propiedad al botón, también se modifica una propiedad de su padre (*FontSize*).

4.2.1.2 App.cs

Su única función es inicializar todos los paquetes a utilizar en la aplicación, en nuestro caso todos los básicos de Xamarin.Forms y el de Syncfusion Calendar; además de llamar a la página principal de la aplicación.

Esta página también alberga la clase del usuario de la aplicación, llamada *MyUser*. La razón de que se haya creado aquí y no en la página principal es debida a la inmutabilidad de *App*. Así, siempre existe la certeza de que las preferencias del usuario serán accesibles en todo momento.

4.2.2 PÁGINA PRINCIPAL: MAINPAGE

La función de la página principal (o *MainPage*) es mostrar de una forma rápida e intuitiva la información más relevante de los eventos del usuario, almacenados en la propiedad *Events* de *MyUser*. Los colores utilizados, que muestran una paleta similar en todas las páginas de la aplicación, han sido seleccionados con el objetivo de contrastar lo máximo posible entre sí. De esta manera, la descripción de los elementos será mucho más sencilla.

Esta página dispone de tres elementos principales, los cuales son, de arriba a abajo: la barra de navegación, el calendario de eventos y la lista de eventos. Debido a que las capturas provienen de un dispositivo Android, la barra inferior se mantiene constante en botones y color a lo largo de toda la aplicación, por lo que no se volverá a hacer mención de ella.

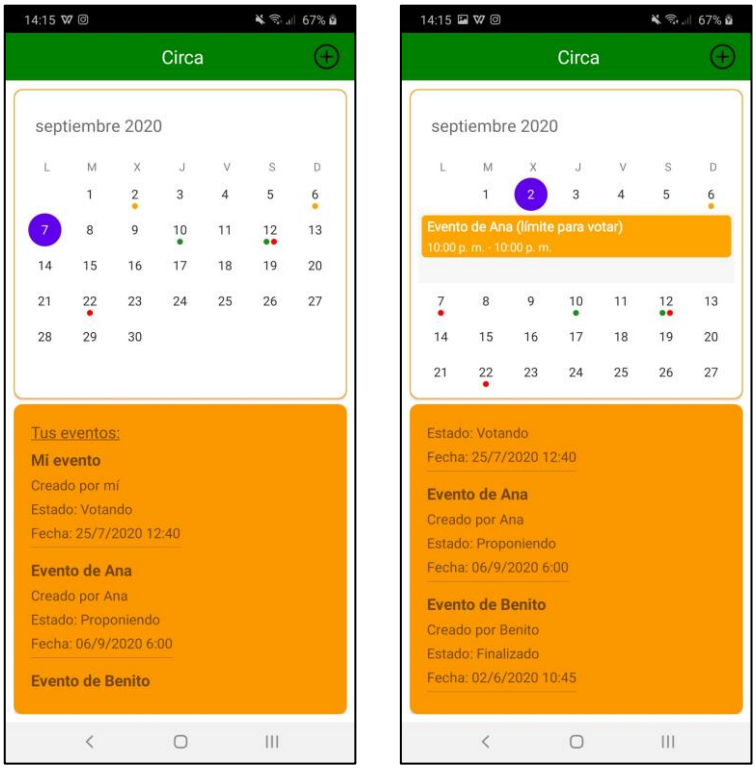

Figura 24. Página principal de Circa

4.2.2.1 La barra de navegación

Mostrada en verde en la parte superior, la barra de navegación —conocida como *NavigationBar* en inglés— es el único elemento constante en todas las pestañas de la aplicación, aunque presenta modificaciones.

Figura 25. Detalle de la página principal, donde se muestra la barra de navegación

En este caso, únicamente muestra el título "Circa" y un *ImageButton*, un elemento con las mismas funciones que un botón con la diferencia de que muestra una imagen en lugar de una cadena de texto. Al pulsar la imagen, se activa el evento asíncrono *NewEventCreation_Clicked*, que llama a una nueva página de creación de evento, de la cual se hablará más adelante.

4.2.2.2 El calendario de eventos

Rodeado con un rectángulo naranja, el calendario es uno de los elementos más importantes y novedosos de la aplicación, ya que permite ver toda la información relativa a las fechas de una forma visual e interactiva. Se trata del widget obtenido del ya mencionado paquete Syncfusion Calendar, el cual lleva el nombre *syncfusion:SfCalendar*. Pese a que hubo complicaciones de implementación durante las primeras semanas de desarrollo, el componente ha demostrado sus altas capacidades de personalización y funcionalidad.

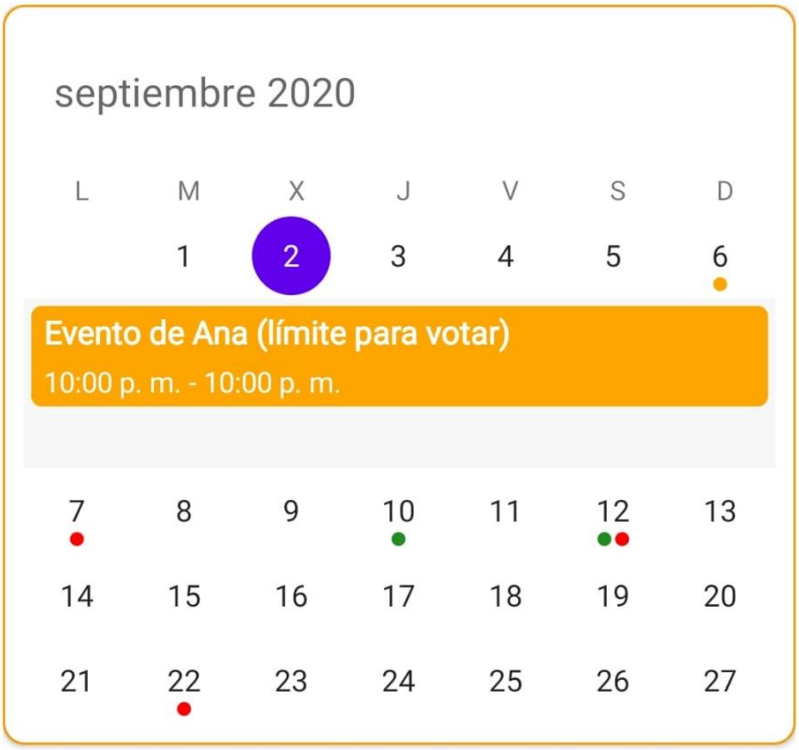

Figura 26. Detalle de la página principal, donde se muestra el calendario de eventos

Respecto a los colores empleados, se ha decidido utilizar un código homogéneo para cada evento, en lugar de emplear un color o conjunto de colores para cada evento. Pese a que este último sistema tiene ciertas ventajas, como una mejor diferenciación de las fechas que pertenecen a cada evento, el tipo de fecha representado sería más difícil de identificar. Los colores de la codificación actual son:

- **Naranja**: Indican la fecha y hora límite para votar. En caso de que el evento permita a los invitados presentar fechas propias, la fecha límite para proponer también se representa de este color.
- **Verde**: Reflejan las fechas propuestas en las cuales nuestro usuario ya ha votado.
- **Rojo**: De forma análoga al verde, muestran las fechas propuestas en las que nuestro usuario no ha votado.

Cabe recordar que, en el estado actual de la aplicación, la interfaz sólo permite establecer un tipo de voto, el cual indica la intención favorable del usuario hacía la fecha propuesta. Pese a ello, el modelo de clases propuesto permite otras opciones, las cuales se discutirán más adelante.

Al pulsar sobre las diferentes fechas, la fila inferior se repliega permitiendo mostrar las fechas "citas" del día. Estas citas poseen un título, una hora de inicio y una hora de finalización. Debido a que la fecha de finalización no se puede dejar en blanco, se ha decidido duplicar la de inicio. Cabe mencionar que actualmente, al pulsar sobre las fechas o los eventos desplegados, no se realiza ninguna acción adicional, por lo que podríamos considerar este elemento como "pasivo" dentro de la página.

4.2.2.3 La lista de eventos

Situada en la parte inferior de la aplicación, la lista de eventos se trata de un elemento *CollectionView*, el cual, mediante un enlace de datos bidireccional, mantiene en constante sincronización los cambios que se producen en los eventos, ya sea por la modificación de uno ya existente o por la creación de uno nuevo. Pese a que lo usual en Xamarin.Forms es emplear el elemento *ListView* para representar listas de datos, *CollectionView* es más flexible, presenta mejor rendimiento y es mucho más personalizable, lo que la convierte en una opción más atractiva.

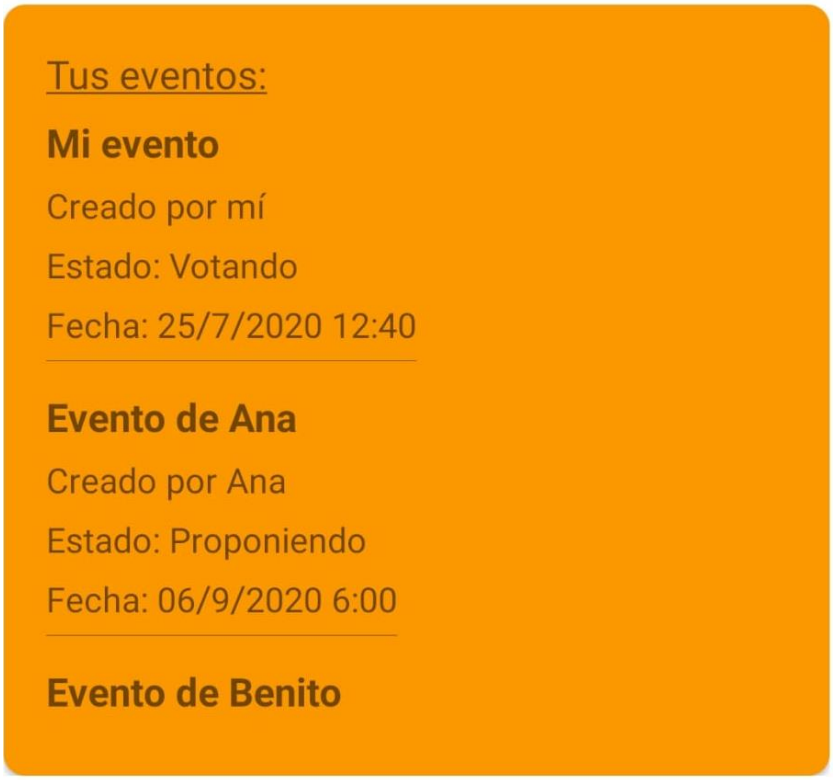

Figura 27. Detalle de la página principal, donde se muestra la lista de eventos

Siguiendo las pautas del patrón MVVM, las propiedades de los eventos no son mostradas directamente, sino que las propiedades son trasladadas a la clase *GenericEventWrapper* a través de un constructor predefinido. Este método, además de mantener separados el modelo y la vista, aligera el procesamiento de datos de la interfaz, ya que los tipos del *wrapper* son nativos de XAML, como cadenas de texto (*string*) y fechas (*Datetime*). De esta manera, la información procesada de cada evento aparece representada en una celda del *CollectionView*. Esta información se corresponde con:

- **Título**: Muestra el título del evento, sin cambios. Para evitar huecos vacíos, si el evento no posee título, la celda mostrará en su lugar la cadena "<Sin título>".
- **Apodo del administrador**: Refleja el nombre por el que es conocido el administrador.
- **Estado (o estatus)**: Indica cuál es la situación actual del evento, que puede ser:

- "Proponiendo", si la fecha límite para proponer no ha vencido. Como es lógico, este estado sólo se puede dar en los eventos en los que los invitados puedan proponer fechas.
- "Votando", si la fecha límite para votar no ha vencido. En caso de haber fecha de proposición, ésta tampoco debe de haber vencido.
- "Finalizado", si la fecha actual es posterior a la de votación.

- **Fecha relevante**: Dependiendo del estatus y la naturaleza del evento, la fecha mostrada será la de votación o la de proposición.

Esta lista, al contrario que el calendario, sí es un elemento activo de interacción con el usuario. Al pulsar sobre cualquiera de las celdas, el sistema recoge el evento escogido y traslada sus datos a una de las páginas de gestión de eventos.

4.2.3 PÁGINA DE EVENTO: *GENERICEVENTPAGE*

La página de evento (o *GenericEventPage*) es una clase C# abstracta que sirve como base de las páginas de gestión de fechas de Circa, que se explicarán más adelante. Se trata de una página de tipo *TabbedPage*, la cual permite anidar otras páginas accesibles por un sencillo desplazamiento lateral. Para evitar confusión en la denominación, de ahora en adelante se llamará a estas páginas anidadas "pestañas".

Todas las páginas de evento poseen dos pestañas. La primera es la de "Características", y engloba todos los campos de datos de *GenericEvent*. Esta pestaña muestra los mismos elementos y funcionalidades independientemente del tipo de evento y su momento de vida (estatus). La segunda es la de "Fechas", la cual presenta dos versiones intrínsicamente diferentes según la configuración inicial del evento y el estatus.

4.2.3.1 Parte general

Todas las páginas de evento, independientemente de sus características, tienen en común una serie de elementos visuales y lógica programática. Para ayudar al usuario a identificarlas y mantener cierta separación lógica, estos elementos comunes se encuentran en la pestaña de "Características". Esta pestaña dispone de tres elementos principales, que son, de arriba a

abajo: la barra de navegación, la barra de desplazamiento y los bloques de información general.

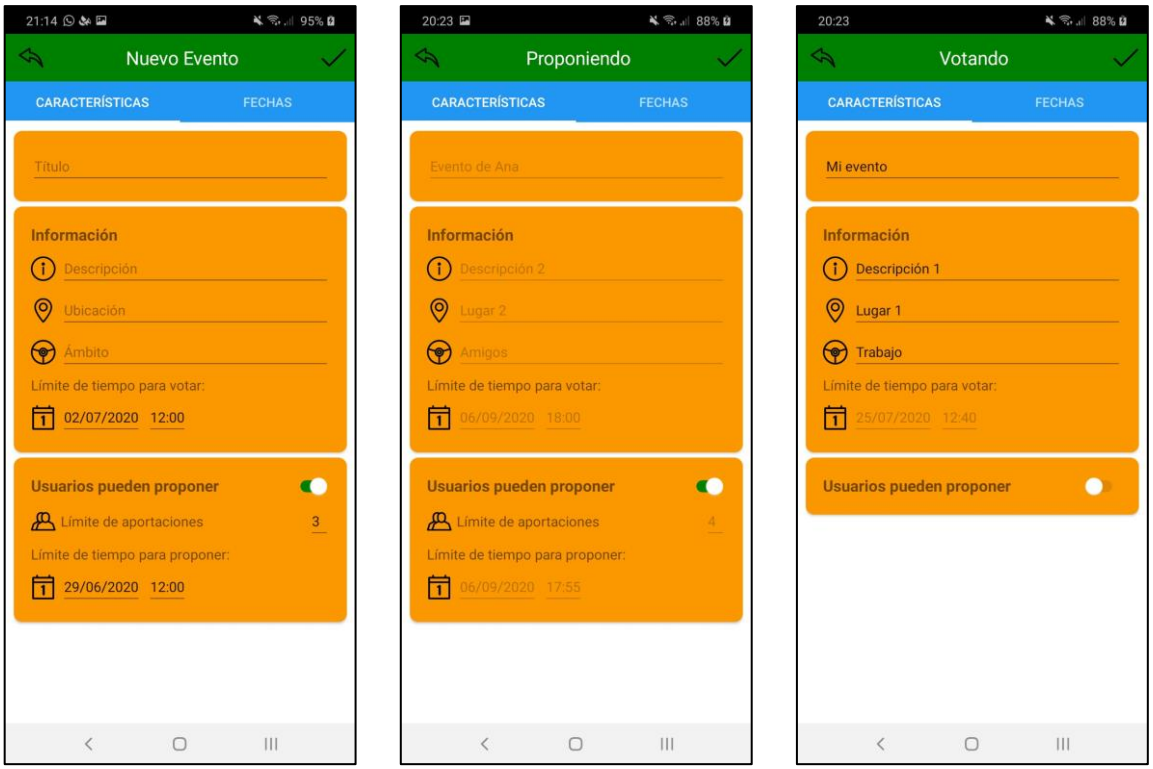

Figura 28. Pestaña de características de la página de evento

4.2.3.1.1 La barra de navegación

Similar a la ya descrita de la página principal, la barra de navegación de la página de evento muestra dos *ImageButton* enfrentados en los extremos con un título en el centro.

Figura 29. Detalle de la página de evento, donde se muestra la barra de navegación

El elemento central sirve para informar al usuario del estatus de la aplicación, mientras que los dos botones sirven para volver a la pantalla inicial, con la diferencia de que el izquierdo no guardar los cambios producidos en el evento, al contrario que el derecho. Esto lo hacen a

través de los eventos asíncronos *CancelEvent_Clicked* y *ConfirmEvent_Clicked*, respectivamente.

4.2.3.1.2 La barra de desplazamiento

La barra de desplazamiento es una característica exclusiva de las páginas TabbedPage que permite al usuario visualizar en que pestaña se encuentra; en este caso la de características o la de gestión de fechas, tituladas "Características" y "Fechas" respectivamente.

Figura 30. Detalle de la página de evento, donde se muestra la barra de desplazamiento

4.2.3.1.3 Los bloques de información general

Este apartado lo forman todos los elementos ubicados en los tres marcos naranjas. Los dos primeros reflejan información común a todos los eventos, mientras que el último contiene los campos relevantes sólo a los eventos en los que los invitados pueden proponer fechas. Todos los elementos de entrada de datos están precedidos por un icono, además de tener un valor predefinido si el campo está vacío. Como se puede apreciar, el título es el único que no posee icono, además de estar en un bloque diferente. Esto es debido a que se quería resaltar su importancia frente al resto.

Para evitar complicaciones de implementación, únicamente los cuatro primeros campos de datos —que se diferencian de los parámetros estáticos por el subrayado— son modificables tras la creación del evento. Esta función únicamente la puede desempeñar el administrador, siempre y cuando la fecha en la que lo haga no sea posterior a la fecha de votación. Esto es debido a que, pasada esta fecha, el evento se da por finalizado y no permite ninguna clase de modificación, ni en esta pestaña ni en "Fechas".

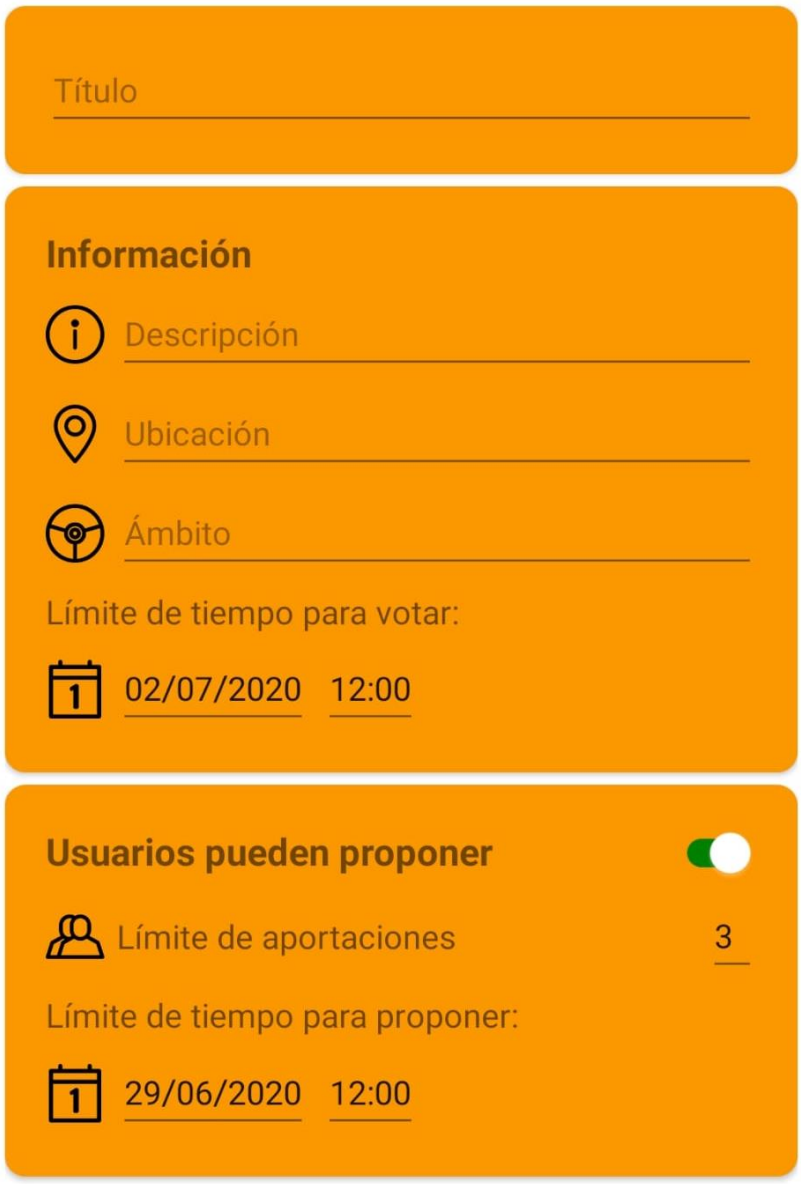

Figura 31. Detalle de la página de evento, donde se muestra la información general

Esta sección incluye diez campos de entrada, los cuales se corresponden con ocho de las nueve propiedades de la clase *GenericEvent* de Circa. Para explicar mejor su funcionalidad, se ha decidió ordenar los campos por tipo en lugar de por su orden en la interfaz:

- **Entradas de texto**: Son elementos *Entry* que permiten introducir cadenas de texto codificadas en UTF-8. Actualmente, todos los campos de texto pueden quedar vacíos y no se hace ninguna clase de comprobación de los caracteres introducidos.

Figura 32. Ejemplo de introducción de datos en un elemento Entry

Las entradas de texto que aparecen en esta pestaña son:

- Título: Ayuda a identificar fácilmente el evento en el listado. Se puede dejar en blanco, aunque no es recomendable, ya que será muy complicado para los invitados y el propio administrador discriminar los eventos entre sí. Se corresponde con la propiedad *Title*.
- Descripción: Aporta detalles adicionales sobre el evento. Similar al título, pero reducido a un segundo plano. Puede dejarse en blanco y se corresponde con la propiedad *Description*.
- Ubicación: Indica el lugar donde el evento va a tener lugar. Puede dejarse en blanco y se corresponde con la propiedad *Ubication*.

- Selectores: Son elementos *Picker* que permiten elegir un elemento de una lista de datos predefinida. Es muy común combinar este elemento con un diccionario clave-valor, mostrando por pantalla el valor y guardando en el *viewmodel* la clave.

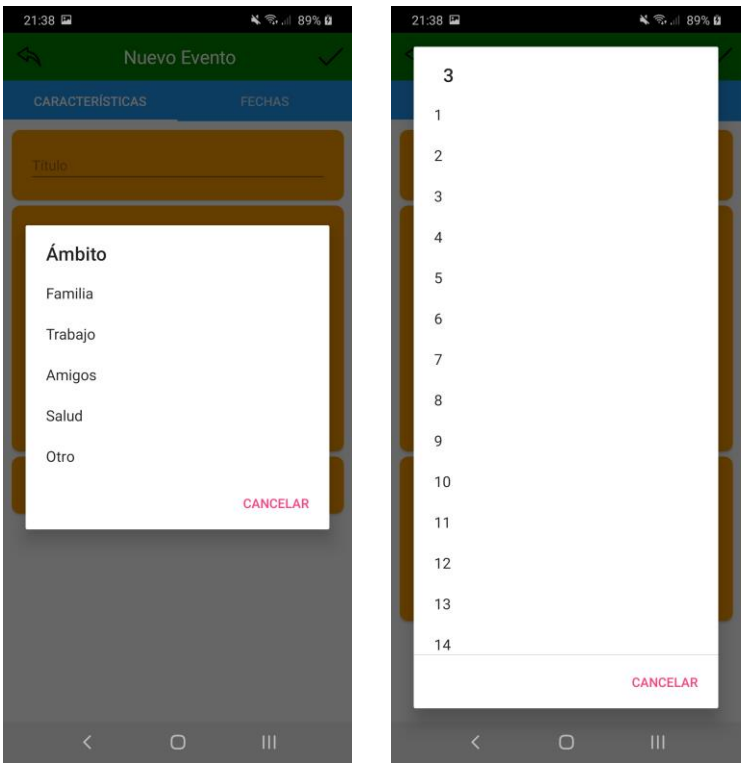

Figura 33. Ejemplos de entrada de datos en elementos Picker

Los selectores que aparecen en esta página son:

- **Ámbito**: Señala el tipo de evento creado. Aunque no es obligatorio, es recomendable rellenarlo, ya que permite agrupar los eventos en conjuntos fácilmente diferenciables, como "Amigos" o "Familia", entre otros. En caso de no seleccionar ninguno de los existentes, el evento se registrará automáticamente en "Otros". Se corresponde con la propiedad *FieldKey*, que almacena la clave del elemento de diccionario seleccionado.
- **Límite de aportaciones por usuario**: Indica el número máximo de fechas que un usuario puede marcar. Puede alcanzar valores entre 1 y 16, ambos incluidos. Su valor predefinido es 3 y se corresponde con la propiedad *MaxPropositionsPerUser*.

- **Selectores de fecha y hora**: Son elementos *DatePicker* y *TimePicker*, respectivamente. Con el objetivo de evitar conflictos lógicos, cada vez que se

cambia cualquiera de estos valores se realizan comprobaciones en el *viewmodel*, que se explicarán más adelante.

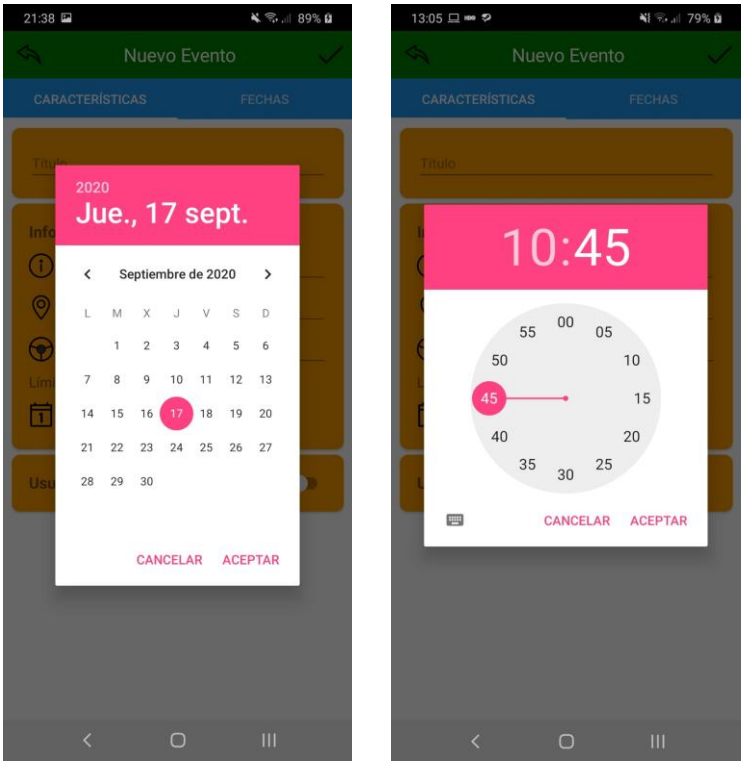

Figura 34. Ejemplos de entrada de datos en elementos DatePicker y TimePicker, respectivamente

Los selectores de fecha y hora que aparecen en esta pestaña son:

  - Fecha y hora de votación: Su combinación genera la propiedad *VotingDeadline*. Su valor predefinido son las doce del mediodía del día correspondiente a una semana después del día de creación.
  - Fecha y hora proposición: Su combinación genera la propiedad *ProposingDeadline*. Su valor predefinido son las doce del mediodía del día correspondiente a cuatro días después del día de creación.

- **Interruptor**: Es un elemento *Switch* que proporciona un valor de estado de alternancia. Su función en el diseño es doble: esconder o mostrar el contenido del último bloque e indicar al *viewmodel* si el evento en cuestión permite o no a los invitados introducir fechas.

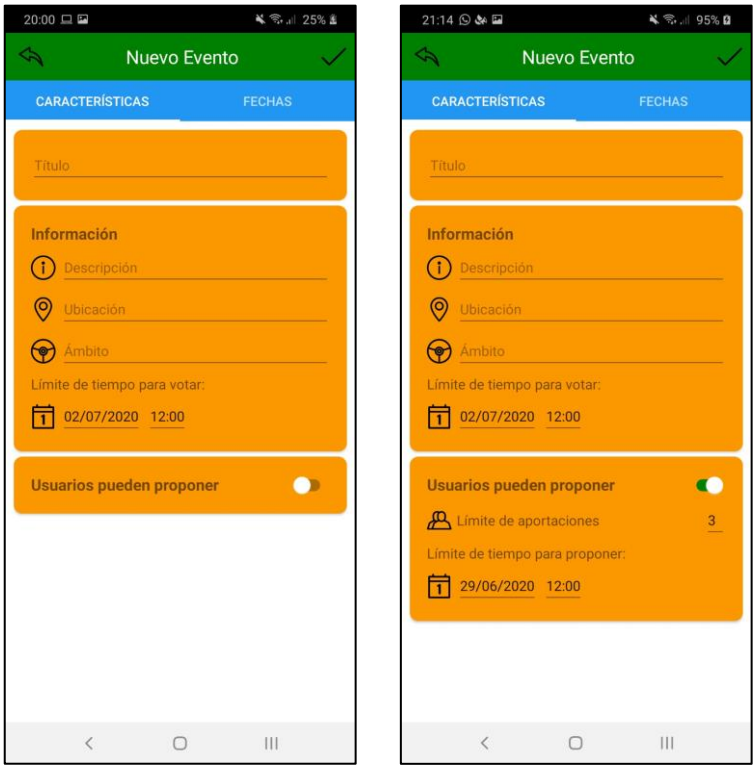

Figura 35. Muestra de los dos estados del Switch

4.2.3.2 Página "Nuevo Evento": ProposingDateEvent I

Pese a su nombre, la página de proposición sirve en realidad para dos momentos de vida del evento: el de creación ("Nuevo evento") y el de propuesta de fechas ("Proponiendo"). La razón de ello es que ambas poseen una gran semejanza estética y funcional, por lo que se ha preferido presentarlas por separado al existir algunas diferencias importantes. Como indica el título de este apartado, en primer lugar, trataremos la página "Nuevo Evento".

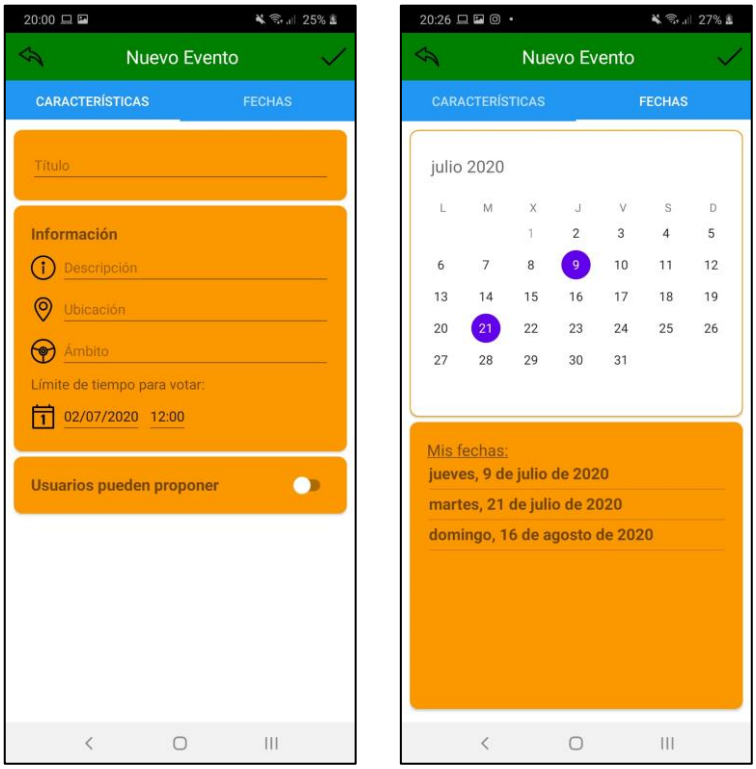

Figura 36. Página "Nuevo Evento", configurada para que sólo el administrador pueda proponer fechas

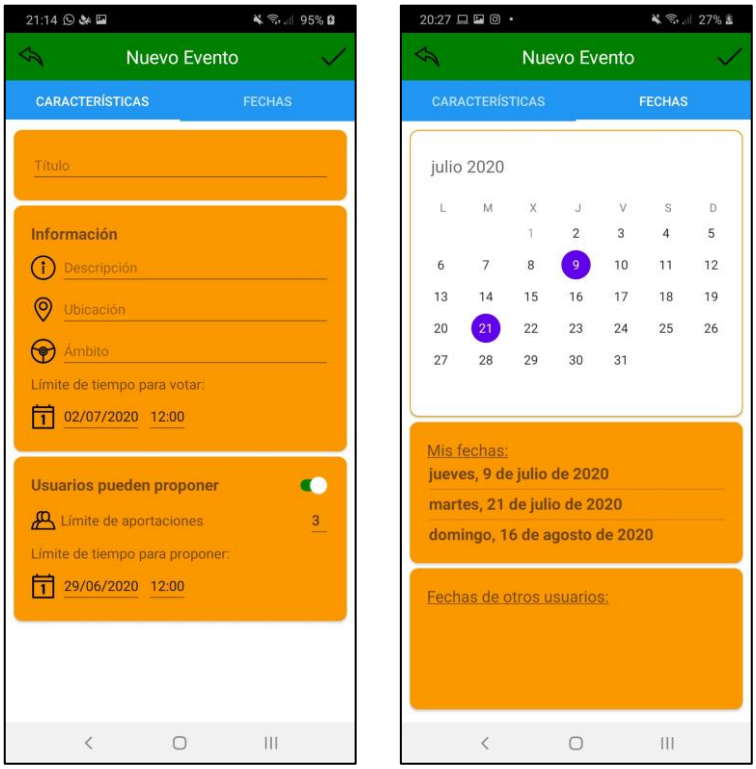

Figura 37. Página "Nuevo Evento", configurada para que los invitados puedan proponer fechas

4.2.3.2.1 El calendario de fechas

Idéntico en estética al de la página principal, el calendario de fechas se trata del elemento activo de la pestaña de fechas, en oposición a la lista, que es íntegramente pasiva. Al contrario que su homólogo, este elemento no sólo desarrolla una función estética u orientativa, sino que se trata del medio otorgado al usuario para introducir sus fechas deseadas. A través de la propiedad *MyDates*, las fechas seleccionadas son enlazadas con el *viewmodel*, donde se comprueba si su número total supera el establecido por el límite de aportaciones por usuario. Esto último, claro está, sólo se realiza si el evento permite a los invitados proponer.

Figura 38. Detalle de la página "Nuevo Evento", donde se muestra el calendario de fechas

Este calendario posee otra propiedad muy importante, *MinDate*, que es la fecha mínima que el usuario, ya sea administrador o no, puede proponer. Su función es esencial, ya que impide que el usuario pueda proponer fechas anteriores al fin de la fecha de votación. Para lograr mantener ambos parámetros sincronizados es necesario enlazar esta fecha con la propiedad *CalendarMinDate* del *viewmodel*, de forma que esta última sea actualizada cada vez que el usuario cambie la primera. Cabe también la posibilidad de que, al cambiar la fecha de votación hacia el futuro, algunas fechas seleccionadas pasen a ser inválidas. Esto se soluciona de forma sencilla eliminando de la selección los elementos no permitidos, además de actualizando la lista inferior para que no muestre las fechas antiguas.

En el ejemplo superior, se puede ver como el primer día del mes está en un tono gris claro, lo que indica que no se puede seleccionar debido a que se encuentra todavía dentro del periodo de votación —es decir, es anterior al 2 de julio a las 12:00—.

4.2.3.2.2 La lista con las fechas del usuario

Se trata de un elemento pasivo con una función clara: mostrar al usuario las fechas que ha seleccionado en el calendario superior. Pese a que no es estrictamente necesario, ya que toda la información que aporta la da el calendario, puede de ser utilidad en situaciones en las que las fechas se encuentren muy alejadas unas de otras.

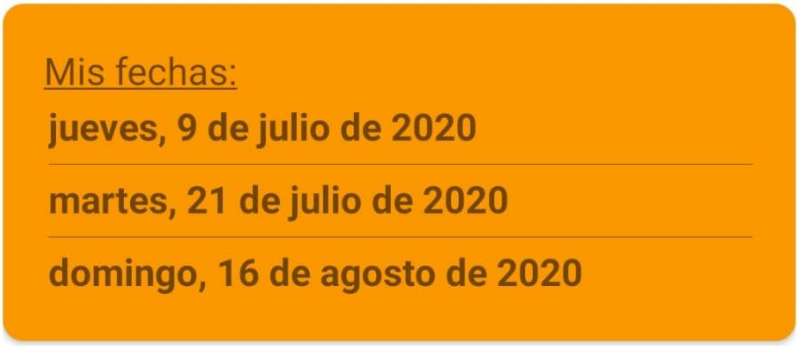

Figura 39. Detalle de la página "Nuevo Evento", donde se muestra la lista con las fechas del usuario

Se actualiza cada vez que se produce un cambio en el calendario, reordenando las fechas, si es necesario, para mantenerlas en constante orden cronológico. Como se puede observar, se ha empleado un estilo "tradicional" para mostrarlas, de forma que al usuario le sean más cómodas de leer.

4.2.3.2.3 La lista con las fechas de los otros usuarios

La función de este elemento, pasivo al igual que la lista anterior, es mostrar las propuestas de los otros usuarios.

Figura 40. Detalle de la página "Nuevo Evento", donde se muestra la lista con las fechas de los otros usuarios

Debido a que en el estado de creación esta lista siempre se encuentra vacía, podría considerarse innecesario mostrarla. Aparece debido a que se ha considerado importante "recordar" al administrador que ha configurado el evento para que otorgue permisos especiales a los invitados. Si este es el caso, esta lista ocupará la mitad inferior del espacio que antes correspondía exclusivamente a la de las fechas del usuario.

4.2.3.3 Página "Proponiendo": ProposingDateEvent II

Estéticamente, la segunda variante de la página *ProposingDateEvent*, "Proponiendo", es idéntica a "Nuevo Evento". En este caso, sus diferencias radican en su funcionalidad interna y la forma en la que presentan los datos. Para que esta página se abra en lugar de otra de gestión de evento deben darse dos condiciones:

1. Los invitados deben poder proponer eventos.
2. La fecha actual debe ser previa a la fecha límite para proponer.

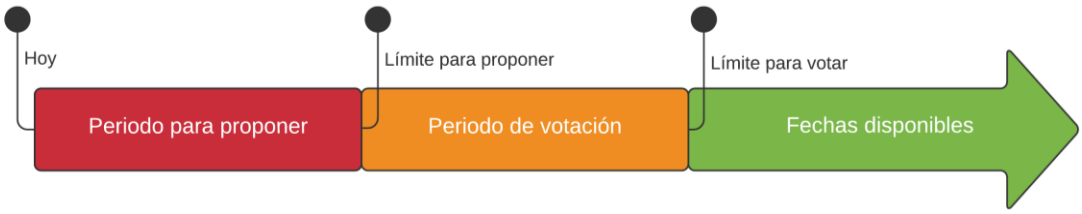

Figura 41. Cronograma de uso estando la opción de proponer habilitada

En caso de que la fecha actual no cumpla la segunda condición, el evento se gestionaría a través de la página "Votando" o "Finalizado", según corresponda.

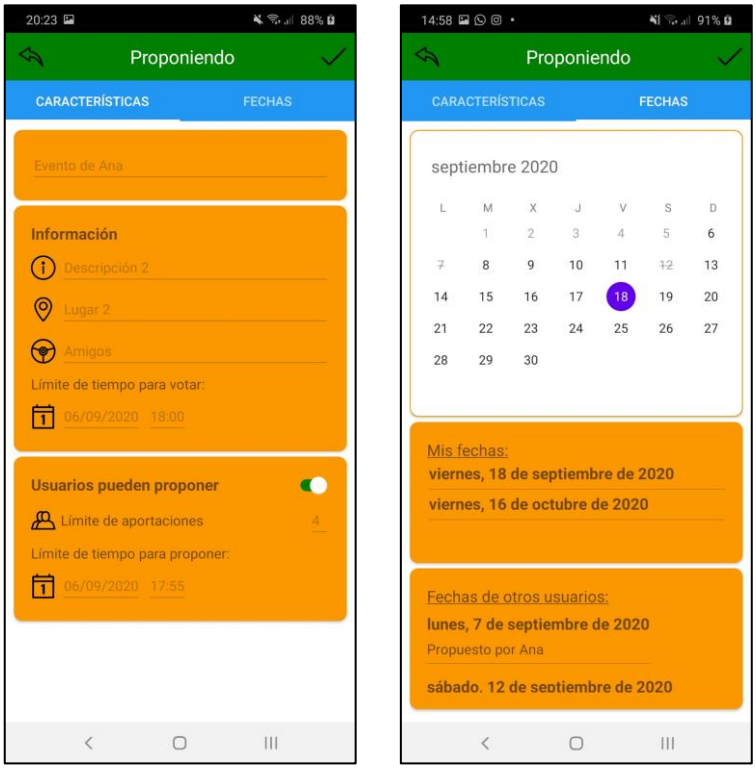

Figura 42. Página "Proponiendo"

Para evitar duplicidades a la hora de describir sus elementos, las descripciones de "Proponiendo" se centrarán exclusivamente en sus diferencias y novedades respecto a la de "Nuevo Evento".

4.2.3.3.1 El calendario de fechas

Como novedad, el calendario de esta página hace uso de la propiedad *BlackoutDates*, la cual permite convenientemente marcar una lista de fechas como no seleccionables. Esta lista se corresponde con las fechas ya seleccionadas por otros usuarios, las cuales se bloquean para evitar duplicidades innecesarias. Esta lista se obtiene directamente del constructor del *viewmodel*, el cual recibe el evento y su correspondiente lista de fechas propuestas. El

sistema discrimina en dos listas, llamadas *MyDates* y *OtherDateOptions*, los eventos propuestos por nuestro usuario y por el resto de los participantes, respectivamente. De igual forma que en "Nuevo Evento", *MyDates* aparece sin ninguna clase de procesamiento previo en la primera lista, mientras que *OtherDateOptions* recibe un tratamiento especial que se explicará más adelante.

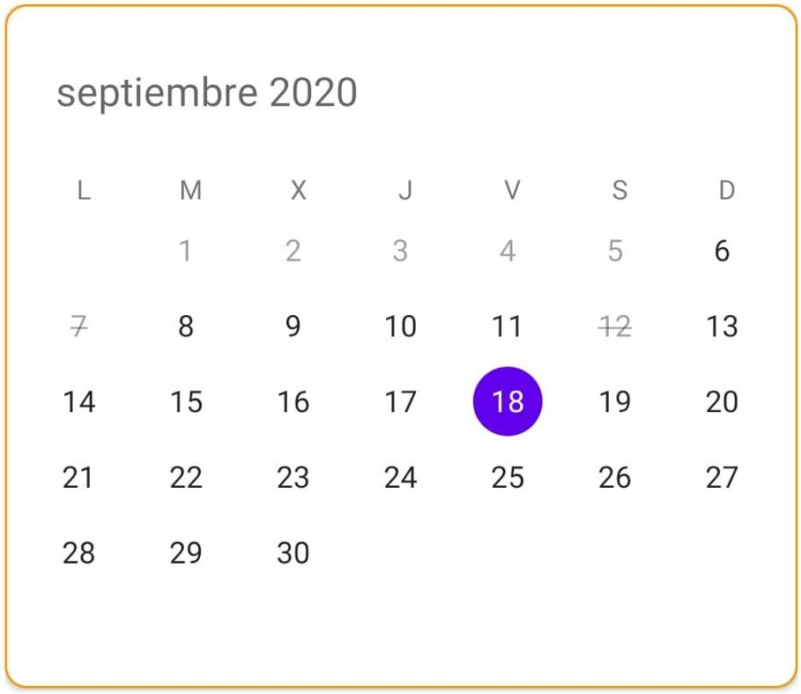

Figura 43. Detalle de la página "Proponiendo", donde se muestra el calendario de fechas

Como se puede apreciar en la figura superior, los días 7 y 12 aparecen tachados y con el mismo tono que las fechas bloqueadas por *MinDate*. Estas son las que se corresponden con las anteriormente mencionadas *BlackOutDates*.

4.2.3.3.2 La lista con las fechas del usuario

No presenta ninguna diferencia, ni funcional ni estética, con "Nuevo Evento".

4.2.3.3.3 La lista con las opciones de los otros usuarios

En esta lista aparece las fechas seleccionadas por usuarios diferentes al nuestro, junto con su apodo. De forma análoga a los eventos de la lista de la página principal, las propiedades deseadas del objeto *DateOption* se trasladan a la clase *DateOptionWrapper*. Debido a que este *wrapper* es usado de nuevo en la página de voto, se explicará con más detalle en el apartado siguiente.

Figura 44. Detalle de la página "Proponiendo", donde se muestra la lista con las opciones de los otros usuarios

Debido a su elevada altura relativa, apenas un solo elemento de la lista puede aparecer al mismo tiempo por pantalla, lo que en muchos casos resulta incómodo para navegar y buscar la información. Por ello se barajaron diseños que fueran capaces de ocultar, mostrar y cambiar el tamaño de las listas y el calendario, de tal manera que estos pudieran ser modificados por el usuario a voluntad, pero debido a la complejidad de su implementación, se decidió mantener un diseño más estático, centrado más en la funcionalidad que en la estética o la facilidad de uso, aunque no se descarta aplicar estas ideas en diseños futuros.

4.2.3.4 Página "Votando": VotingDateEvent I

De forma análoga a la página de proposición (*ProposingDateEventPage*), la página de voto sirve a dos momentos de vida del evento: el de voto ("Votando") y el de finalizado ("Finalizado"). En este caso, la página reservada al evento finalizado tan sólo contiene diferencias menores respecto a la de voto, por lo que no se hará un análisis completo de los elementos de ésta como en los apartados anteriores.

Para que esta página se abra es necesario que la fecha actual esté dentro del periodo de votación, independientemente de si el evento permite o no a los invitados proponer fechas. Esto se traduce en que la fecha en la que se accede a la página de gestión del evento debe ser anterior a la fecha límite de voto, y posterior al vencimiento de la fecha límite de proposición, si el evento lo requiere.

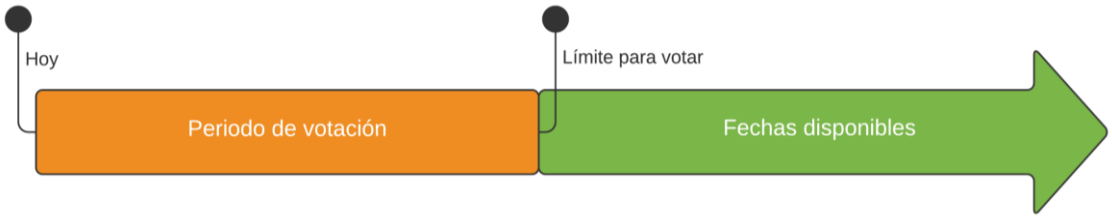

Figura 45. Cronograma de uso estando la opción de proponer deshabilitada

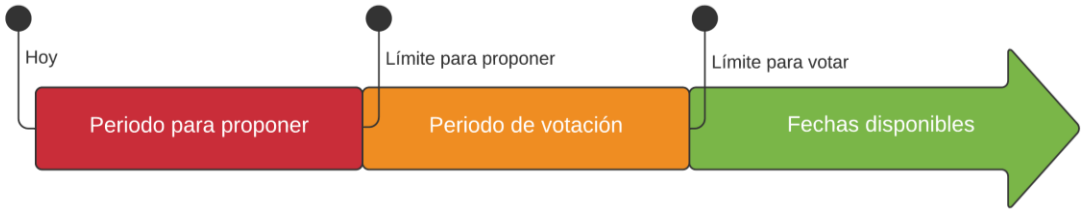

Figura 46. Cronograma de uso estando la opción de proponer habilitada

En caso de que la fecha actual sea posterior al periodo de votación, el evento se dará por finalizado y éste será gestionado por la página homónima.

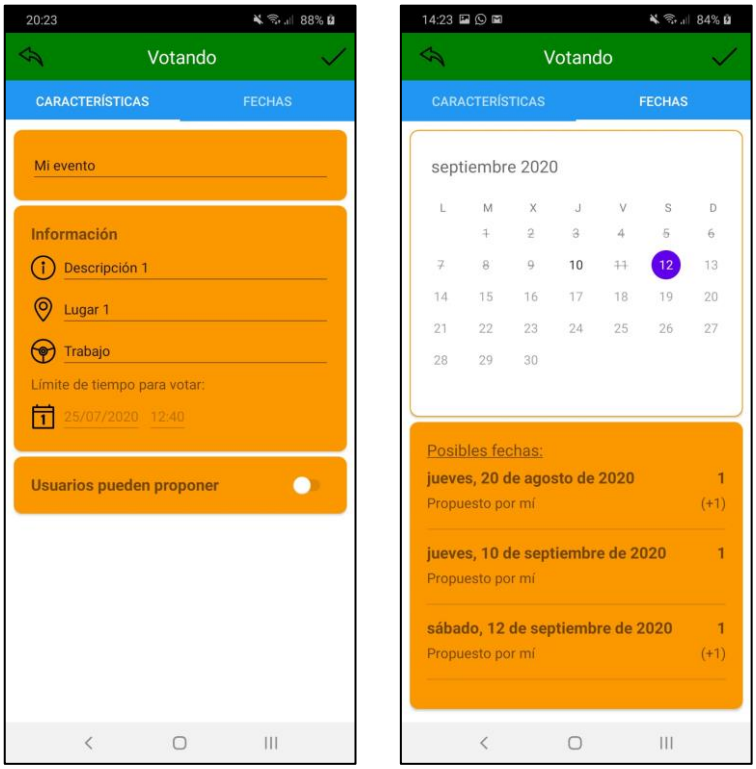

Figura 47. Página "Votando", configurada para que sólo el administrador pueda proponer fechas

La figura superior es un buen ejemplo de cómo el administrador tiene la posibilidad de modificar los cuatro primeros campos de datos en cualquier momento de vida del evento. Como se puede apreciar, estos elementos de entrada no se encuentran bloqueados, por lo que se pueden modificar como si de una página de creación se tratara.

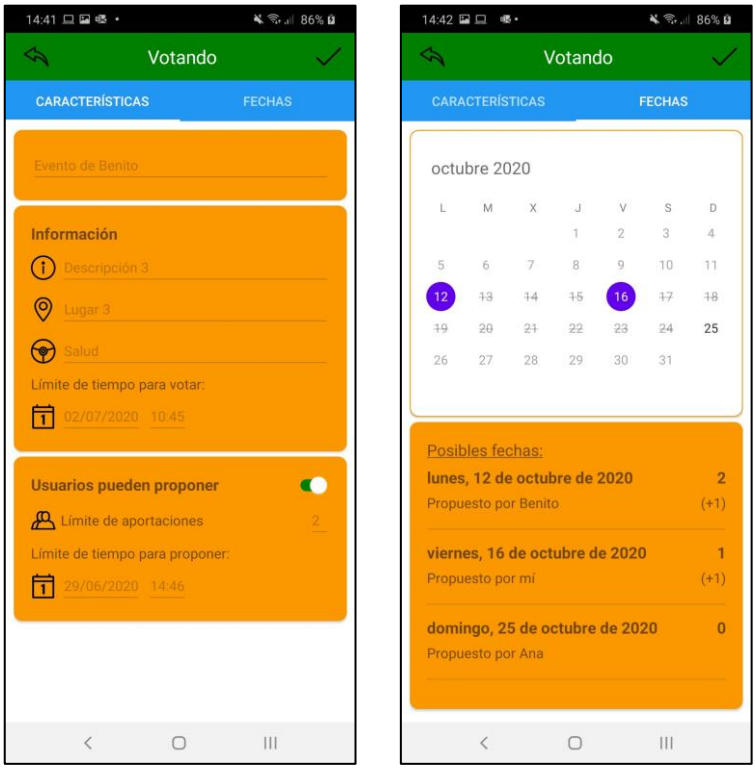

Figura 48. Página "Votando", configurada para que los invitados puedan proponer fechas

La página "Votando" tiene dos elementos principales: el calendario de votos y la lista de opciones.

4.2.3.4.1 El calendario de votos

Como su nombre indica, el calendario de votación es la herramienta de la que dispone el usuario para votar las fechas propuestas por los participantes, ya sean exclusivamente propuestas por el administrador o no. Por ello, y comparado con los empleados anteriormente, este calendario es el más restrictivo respecto a lo que libertad de selección se refiere.

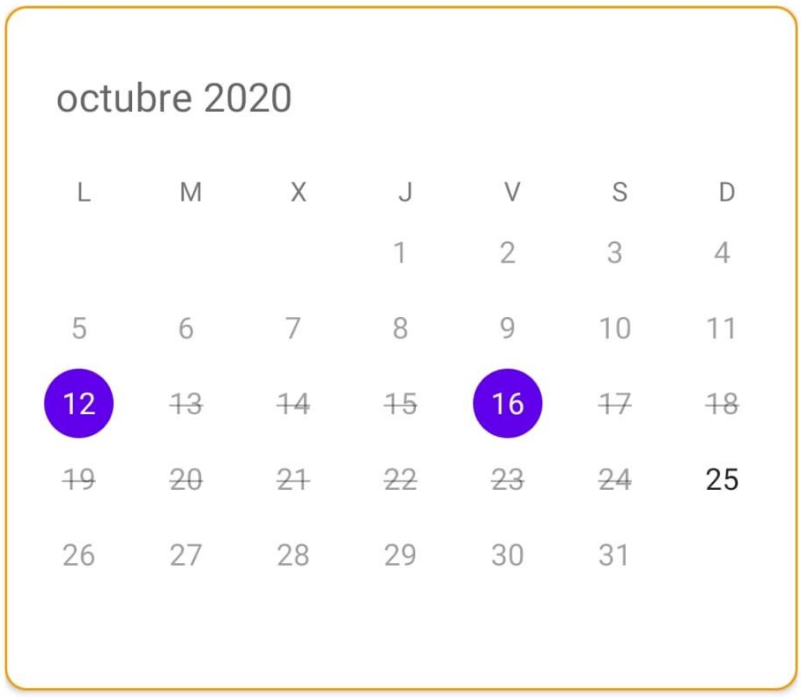

Figura 49. Detalle de la página "Votando", donde se muestra el calendario de votos

En este caso se ha hecho uso de las dos propiedades ya descritas del widget, *MinDate* y *BlackoutDates*, además de una nueva, *MaxDate*. Todas ellas se emplean de forma muy diferente respecto a los casos previos, por lo que se describirá como se ha configurado cada una por separado:

- ***MinDate***: Indica la fecha más antigua que el usuario puede seleccionar. Su valor coincide con la primera fecha de la lista de opciones disponibles —se guardan en orden cronológico—.
- ***MaxDate***: Análoga la *MinDate*, indica la fecha seleccionable más alejada en el tiempo. Su valor coindice con la última de la lista de opciones disponibles.
- ***BlackoutDates***: Se trata de una lista de fechas no seleccionables por el usuario. Para crearse, se ha empleado la función *CreateBlackoutDates*, la cual almacena en una lista todas las fechas desde MinDate hasta MaxDate, para luego eliminar los días propuestos por los usuarios. De esta forma, al ser empleada junto con las propiedades anteriores, únicamente las fechas fuera de la lista son seleccionables.

Lo cierto es que es mucho más intuitivo bloquear todas las fechas excepto unas seleccionadas —una especie de lista "blanca"—, pero curiosamente esta opción no está disponible en el calendario Syncfusion.

4.2.3.4.2 La lista de opciones

La lista de opciones constituye una versión ampliada de la lista de "Proponiendo", en la cual se muestra información adicional sobre el número de votos favorables obtenido en cada evento. La información de cada celda es generada a través del constructor de la clase *DateOptionWrapper*, el cual obtiene y procesa los campos deseados de cada *DateOption* de la lista.

Figura 50. Detalle de la página "Votando", donde se muestra la lista con las opciones de los usuarios

La información mostrada en cada celda se corresponde con los siguientes campos:

- **Fecha**: Muestra la fecha del evento en formato largo.

- ***Nickname***: Refleja el nombre del participante que ha propuesto la fecha, precedido por la frase "Propuesto por". En caso de que la fecha fuera nuestra, este valor será sustituido por "mí" para reflejar mayor naturalidad.
- **Contador de votos**: Almacena el número de votos de todos los usuarios, menos el propio. Se obtiene a través de la función *CountOtherVotes*, la cual contabiliza el número total de personas que han votado el evento excluyendo a nuestro usuario, si es que ha votado.
- **"(+1)"**: Se trata de un *Label* de texto estático, el cual sólo es visible si nuestro usuario ha seleccionado esa fecha en el calendario. La visibilidad del *Label* es regulada a través de la propiedad *MyUserHasVoted* del *wrapper*, la cual se actualiza de forma asíncrona cada vez que la selección del calendario —la propiedad *MyVotes* del *viewmodel*— varía.

Al contrario que las listas de "Proponiendo", la presentada en este apartado es estrictamente necesaria, ya que es el único elemento que aporta en número de votos de cada opción, que es en definitiva el objetivo de la aplicación. Si bien es cierto que sería más sencillo contar directamente el número de votos totales en lugar de emplear dos propiedades diferentes, se ha considerado que de esta manera el usuario tiene presente las opciones escogidas sin necesidad de mirar el calendario.

4.2.3.5 Página "Finalizado": VotingDateEvent II

Como se ha mencionó en el apartado anterior, para que un evento se gestione a través de la página "Finalizado", es necesario que el periodo de votación haya expirado, lo cual ocurre cuando la fecha actual es posterior a la fecha límite de voto (propiedad *VotingDeadline*).

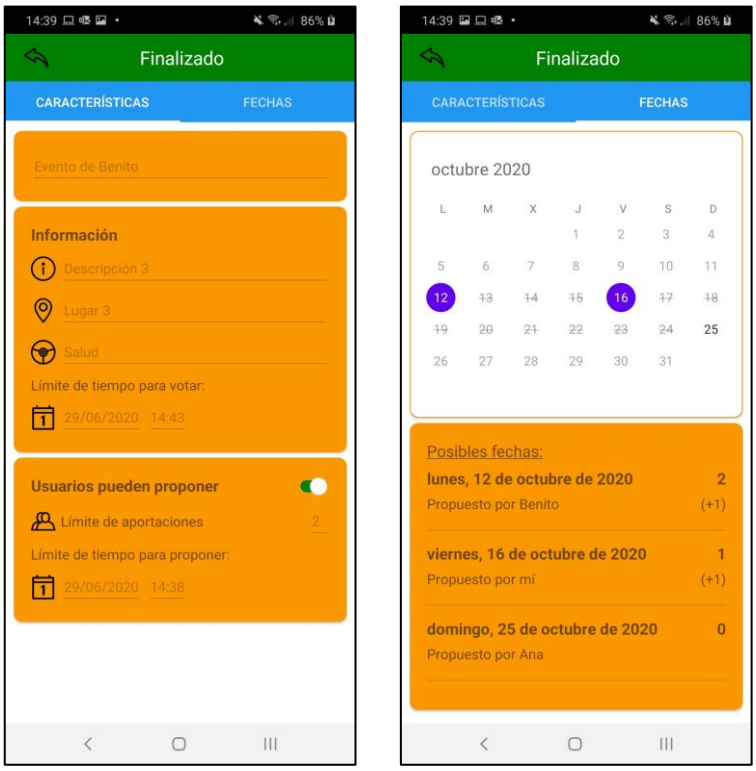

Figura 51. Página "Finalizado"

Como se puede observar en la figura superior, el icono para confirmar los cambios se ha ocultado, de forma que el usuario que accede a la página tiene claro que ya no se pueden realizar cambios en la configuración o en las fechas. Esta es una decisión estética, ya que aun así se han bloqueado los controles del calendario —el único elemento activo de la pestaña "Fechas"—, lo que evita que se cambie la selección realizada por nuestro usuario. Además, se han revocado los poderes especiales del administrador, los cuales desbloqueaban los cuatro primeros campos de la pestaña "Características".

Capítulo 5. INTEGRACIÓN DEL SISTEMA

Tras crear el evento, ya permita o no proponer fechas, es necesario conseguir que el administrador pueda compartirlo con los usuarios que desee. Para lograrlo, es estrictamente necesario el uso de un sistema de acceso compartido que permita a todos los usuarios involucrados visualizar la información del evento y ejercer sus derechos de voto y propuesta según convenga. Este sistema requerirá, como mínimo, de un servidor donde se almacenen todos los datos pertinentes del evento, ya sean los parámetros constantes, como el título o la descripción, o los variables, como son las fechas o los votos. Por ende, el sistema debe ser exógeno a los dispositivos de los usuarios y ser capaz de recibir peticiones constantes de los usuarios de la aplicación. Si bien es cierto que no se requiere un acceso inmediato de los datos o una sincronización perfecta, el sistema no debe tampoco permitir que los datos puedan estar desactualizados durante mucho tiempo.

Este no es un problema poco común en las aplicaciones móviles, ya sean *schedulers*, calendarios electrónicos o de cualquier otro ámbito. Diversas soluciones se han planteado y estudiado para este proyecto, aunque ninguna se ha llegado a implementar de forma efectiva.

5.1 SERVICIO DEDICADO

La solución más simple y tal vez más fácil de implementar es utilizar un servidor dedicado que soporte una base de datos relacional capaz de guardar todos los ficheros y parámetros necesarios para asegurar la funcionalidad buscada. Esta opción ofrecería un gran control sobre la forma en la que se gestionan los campos de datos, además de permitir modificaciones en las relaciones entre las bases de datos y sus atributos, si fueran necesarias.

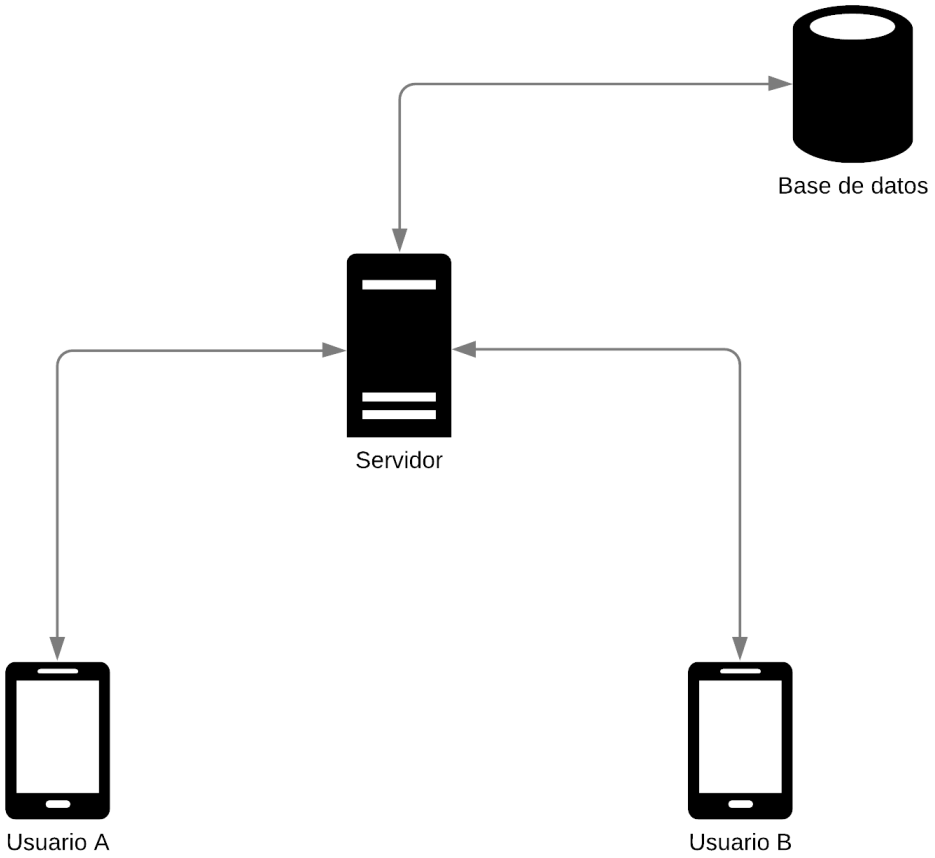

Figura 52. Diagrama simplificado de comunicación entre usuarios con un servidor dedicado

5.1.1 LA BASE DE DATOS DEDICADA

La base de datos mostrada en el diagrama es esencial, ya que, como se constató en la Arquitectura del sistema, es el elemento encargado de guardar toda la información relacionada con los usuarios y sus eventos.

El diagrama mostrado a continuación refleja un posible diseño que podría tomar la base de datos relacional de la aplicación en el lado del servidor, el cual debería ser capaz de gestionar todas las peticiones necesarias sin necesidad de servicios adicionales. La excepción se encuentra en el modelo de la tabla del usuario, que se trata de un *dummy* que únicamente posee un identificador único y un apodo (*nickname* en inglés). Se trata de un reflejo del objeto *AppUser* presente en el modelo de clases.

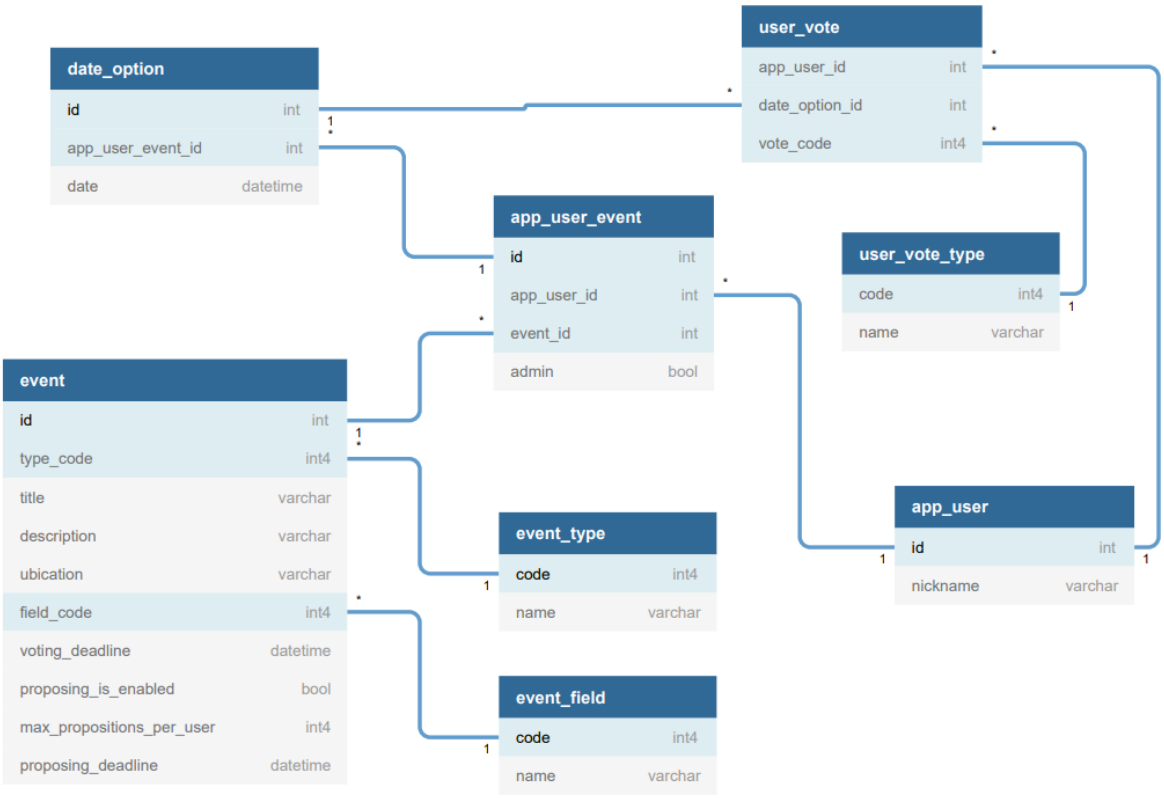

Figura 53. Base de datos relacional para servidor dedicado

Como se puede apreciar, los parámetros de las tablas principales coinciden con los de las clases del modelo de negocio (Modelo):

Tablas principales de la BD relacional	*Clases del modelo*
app_user	AppUser
event	GenericEvent
date_option	DateOption
user_vote	UserVote

Tabla 2. Comparativa entre las tablas principales de la BD y las clases del modelo

Por su parte, las tablas auxiliares se corresponden con los diccionarios (colecciones de clases y valores) existentes en las clases del modelo *GenericEvent* y *UserVote*, respectivamente. Esto permite un ahorro considerable de bits, ya que se elimina la necesidad de repetir cadenas de caracteres en las tablas principales a través de un sistema de clave (*integer*) y valor (*varchar*).

Tablas auxiliares de la BD relacional	*Diccionarios<int, string> del modelo*
event_field	EVENT_FIELDS
user_vote	VOTE_OPTIONS

Tabla 3. Comparativa entre las tablas auxiliares de la BD y los diccionarios del modelo

Además de las ya mencionadas, hay otras dos tablas auxiliares: *event_type* y *app_user_event*. La primera es una tabla código-valor que refleja el tipo de opciones con las que trabaja el evento, y actualmente no tiene ninguna función. Su utilidad futura en la aplicación se discutirá en un apartado posterior. La segunda es una tabla intermedia que sirve para relacionar las tablas de *event* y *app_user*. Su existencia elimina la necesidad de crear una tupla en *event* por cada usuario que administre o sea invitado al evento, eliminando así duplicidades innecesarias. La tabla *app_user_event* tiene la función, además, de indicar que usuario es el administrador del evento en cuestión a través del atributo *admin*. Como se puede apreciar, esta disposición deja abierta la posibilidad de que en el futuro varios usuarios sean administradores del evento sin necesidad de hacer cambios en el modelo relacional.

Pese a sus puntos a favor, existen desventajas a este tipo de arquitectura. En primer lugar, los servidores de hosting suelen tener unos costes relativos a la capacidad de almacenamiento. Aunque la información de los eventos y sus fechas y votos asociados no ocupen mucho espacio en memoria tomados de forma individual, el conjunto agregado de todos ellos podría alcanzar valores muy elevados a lo largo del tiempo si existe un uso continuado de la aplicación.

En segundo lugar, existe el problema de la información y validación de los usuarios. No sólo se tendría que generar un sistema dedicado para crear y autenticar a los usuarios a nivel de aplicación, sino que el servidor deberá mantener su información en la base de datos, garantizando la integridad de sus credenciales y ocupando espacio en memoria.

5.2 USO DE SERVICIOS EXTERNOS

Otra posibilidad que se ha barajado es recurrir a servidores en la nube de acceso gratuito para guardar la información. En lugar de tratar con un sistema dedicado centralizado, se podrían utilizar las funciones ofrecidas por las API de aplicaciones de calendario electrónico, las cuales ya disponen de servicios y sistemas dedicados a optimizar el tratamiento y el guardado de fechas. Grandes compañías tecnológicas, como las ya mencionadas Google y Microsoft, ofrecen estas API a los desarrolladores independientes de forma gratuita, siempre y cuando se cumplan sus requisitos. Al emplear sus servicios, podemos asociar la información de los eventos y las fechas directamente a los ya existentes de los calendarios electrónicos, con los cuales comparten muchas similitudes.

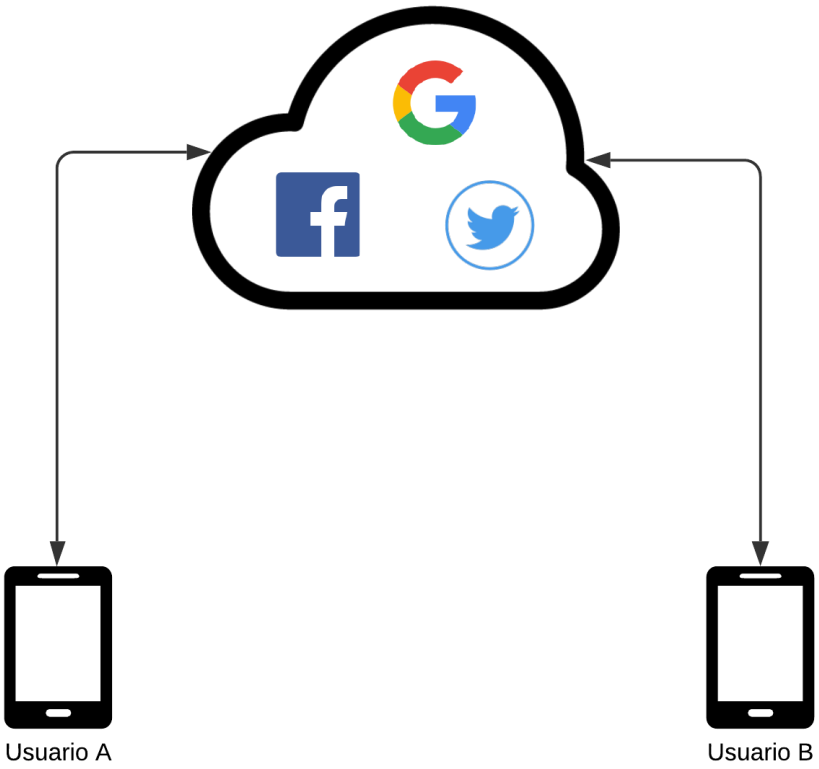

Figura 54. Diagrama simplificado de comunicación entre usuarios empleando sistemas externos

5.2.1 AUTENTICACIÓN DE USUARIOS

Esta opción trae también otra ventaja muy atractiva, que es utilizar su sistema nativo de identificación, gestión y almacenamiento de usuarios. Esta tendencia es una de las más seguidas en el ámbito de la creación y gestión de perfiles del usuario, debido a diversos factores:

- Desde el punto de vista del usuario:
    - La creación de nuevas cuentas sería mucho más rápida, ya que solamente requiere introducir el correo electrónico y la contraseña de la cuenta del servicio externo; por ejemplo, si se utilizaran los servicios de Google, se requeriría un correo de Gmail y su contraseña asociada.
    - Da una mayor sensación de seguridad, debido a que suele existir cierta reticencia a la hora de dar información personal a través de aplicaciones de origen desconocido.

  - No requiere forzar a recordar el correo y la contraseña empleados en la aplicación.

- Desde el punto de vista de los gestores de la aplicación:
  - Muchas compañías han introducido servicios de *backend* e interfaz de usuario para que los desarrolladores puedan emplear sus perfiles de usuario de una manera fácil y segura. Entre los SDK (*software development kit*, o kit de desarrollo de software en español) móviles más empleados para esta tarea se encuentra Firebase Authentication, el cual permite acceder fácilmente iniciar sesión con cuentas de grandes empresas, entre las que se encuentran Google, Facebook y Twitter [31].
  - No requiere disponer de un servidor propio que se encargue de las cuentas de la aplicación. Suelen tratarse de sistemas muy seguros y fiables con un gran renombre en el mundo empresarial.
  - El emplear estas cuentas tiene beneficios adicionales, como el acceso (siempre que sea permitido) a datos y servicios ofrecidos por las API.

Multitud de aplicaciones móviles emplean este tipo de servicios para mejorar la experiencia de usuario, como es el caso de Doodle, que permite, además de un servicio de creación de usuarios propio, utilizar la identificación de cuenta de Facebook o Google [11]. Si se emplean estos servicios, la aplicación ofrece, además, la posibilidad de sincronizar sus calendarios con el suyo propio. Esto permite al usuario visualizar, de una forma natural y completamente integrada con la interfaz, todos los eventos de la aplicación junto con los de su cuenta escogida.

Figura 55. Venta de inicio de sesión de Doodle, en su versión Android

5.2.2 INTEGRACIÓN CON GOOGLE CALENDAR ("CALENDAR")

Desde este momento vamos a tratar un caso hipotético en el cual la tecnología de integración en la nube sea la proporcionada por Google. Es uno de los mejores candidatos, ya que su correo electrónico es de los más utilizados del mundo —en 2018 ya contaba con más de 1.5 billones de usuarios a nivel mundial [32]—, es totalmente gratuito a nivel personal y abre las puertas a una potente API, la cual permite acceder muchos de sus servicios asociados más importantes, como la gestión de cuentas o la de calendarios.

Si partimos de la premisa de que el control de usuarios puede ser totalmente integrado con el sistema en la nube de Google a través de algún SDK, como el ya mencionado Firebase, únicamente quedaría por concretar como se podrían almacenar de los eventos y los votos a

través de la nube y cómo debería hacerse. Para ello nos basaremos en dos elementos fundamentales de Google Calendar: los calendarios [33] y los eventos [34].

Como se mencionó en apartado del Estado del arte, una de las características de Calendar es que a través de su API es posible no sólo crear, gestionar y compartir los eventos, sino también calendarios anidados. Si se piensa de una forma analítica, los eventos de Circa están compuestos por una serie de fechas y horas, algunas de carácter obligatorio, como la fecha límite de voto, y otras de carácter opcional, como las fechas a votar. Una forma eficaz de guardar todos estos parámetros sería crear un calendario compartido entre todos los usuarios del evento de Circa, y asociar a él todos los eventos (de Calendar) necesarios para guardar las fechas y demás información necesaria.

5.2.2.1 Identificación de los objetos de Circa con los de Google Calendar

En primer lugar, trataremos la migración de la información de la clase *GenericEvent* de Circa a los servidores de Google Calendar. De acuerdo con su API oficial para .NET [35], se ha realizado una distribución de los parámetros a través de un calendario compartido, con el nombre de **"Event_calendar"**, y tres eventos adscritos a él:

- **"Generic_info"**: Guarda todos los parámetros generales de *GenericEvent*, exceptuando la fecha límite para votar. Su fecha de creación coincide con la del evento de Circa.
- **"Voting_info"**: Almacena la fecha límite para votar.
- **"Proposing_info"**: Guardará la fecha límite para proponer eventos, además del número máximo de eventos por usuario. Su propia existencia es el indicativo de que el evento permite proponer fechas a los invitados.

5.2.2.1.1 Eventos

La tabla inferior refleja cómo sería una posible interacción entre los parámetros de *GenericEvent* de Circa y los del calendario compartido ("Calendario") y los eventos mencionadas previamente:

Propiedad de Circa		*Propiedad de las clases de la API de Calendar*	
Propiedad de GenericEvent	*Tipo (C#)*	*Propiedad de Calendar y Event*	*Tipo (C#)*
Id	int	"Event_calendar".Id	string
Admin.Id	int	"Generic_info".Creator.Id	string
Admin.Nickname	string	"Generic_info".Creator.DisplayName	string
Title	string	"Generic_info".Summary	string
Description	string	"Generic_info".Description	string
Ubication	string	"Generic_info".Location	string
FieldKey	int	"Generic_info".ExtendedProperty(0)	string
VotingDeadline	DateTime	"Voting_info".Start.DateTime	DateTime
ProposingIsEnabled	bool	-	-
MaxPropositionsPerUser	int	"Proposing_info".Description	string
ProposingDeadline	DateTime	"Proposing_deadline".Start.DateTime	DateTime

Tabla 4. Traspaso de las propiedades de GenericEvent a las clases de Google Calendar

Todas las propiedades han sido asignadas, en lo posible, a campos ya existentes de los calendarios y eventos de Calendar, con una excepción: *FieldKey*, que ha sido asignado a una propiedad extendida (*ExtendedProperty*), de cuya naturaleza se hablará más adelante en el apartado de Escalabilidad. Otro apunte importante es que hay algunos campos que requerirían cambios de tipo, especialmente los ID, ya sean de usuarios o de calendarios. Esto es debido a que los ID de Google se encuentran codificados en Base32, el cual incluye números y letras, lo que podría conllevar modificaciones de algunos tipos del modelo.

5.2.2.1.2 Opciones

En segundo lugar, quedaría establecer un sistema que guarde de forma fiable las posibles fechas. Esto es mucho más sencillo, ya que toda la información puede guardarse en una sola clase *Event* de Calendar, como se puede apreciar en la tabla siguiente:

Propiedad de Circa		*Propiedad de las clases de la API de Calendar*	
Propiedad de DateOption	*Tipo (C#)*	*Propiedad de Event*	*Tipo (C#)*
Id	int	Id	string
Proposer.Id	int	Creator.Id	string
Proposer.Nickname	string	Creator.DisplayName	string
Date	DateTime	Start.DateTime	DateTime

Tabla 5. Traspaso de parámetros de DateEvent a la clase Event de Google Calendar

5.2.2.1.3 Votos

Para finalizar, el sistema de voto se integrará con el sistema ya existente de invitación de la clase *Event* de Calendar, el cual coincide perfectamente en la utilidad buscada. Este sistema es accesible a través de la propiedad *Attendees*, que es una lista que engloba todos los parámetros sobre los invitados del evento y las decisiones que han tomado sobre los mismos. El guardado de los parámetros de *DateOption* se realiza según la siguiente tabla:

Parámetros de Circa		*Clases de la API de Google Calendar*	
Propiedad de DateOption	*Tipo (C#)*	*Propiedad de Event*	*Tipo (C#)*
Voter.Id	int	Id	string

Voter.Nickname	string	DisplayName	string
TypeKey	int	ResponseStatus	string

Tabla 6. Traspaso de propiedades de UserVote a las propiedades de la clase Attendee de Google Calendar

Cabe mencionar que, al emplear el sistema de invitaciones de Google, es necesario realizar una sencilla conversión antes de guardar la propiedad *TypeKey* en *ResponseStatus*. Este ajuste se realizaría según la tabla inferior:

Posibles valores de TypeKey	*Posibles valores de ResponseStatus*
<*No ha votado todavía,* AppVote *no existe*>	"needsAction"
[300] = "No me viene bien"	"declined"
[200] = "No me viene bien, pero podría asistir"	"tentative"
[100] = "Me viene bien"	"accepted"

Tabla 7. Traspaso de los valores de TypeKey a los de ResponseStatus

Con todos estos ajustes, el calendario compartido que hace referencia a cada evento de Circa tendría este aspecto:

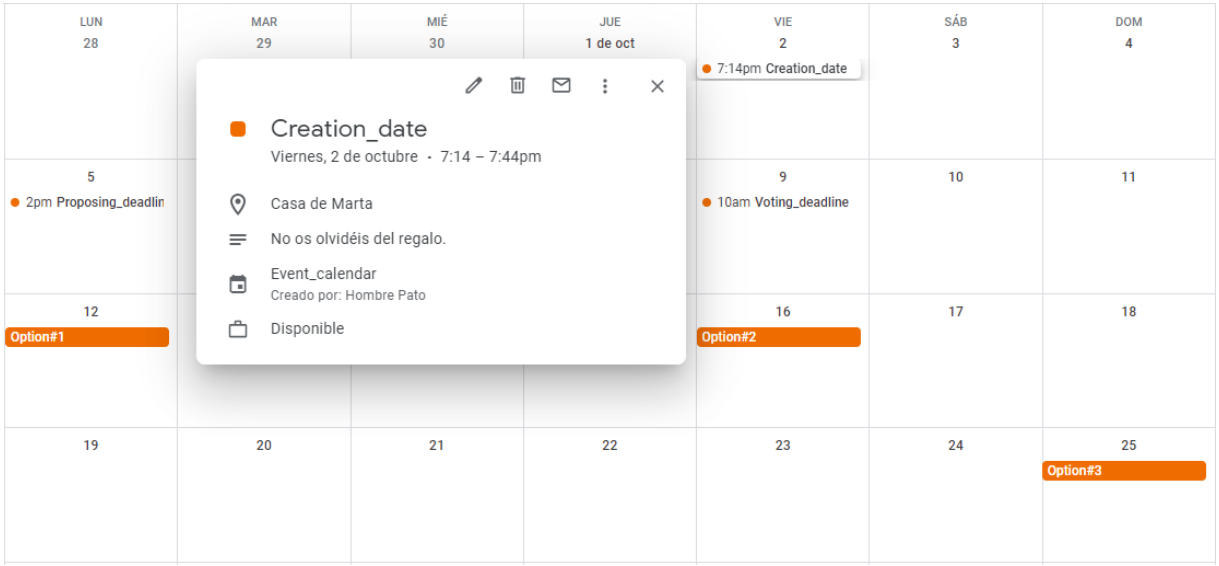

Figura 56. Vista mensual de Google Calendar, donde se aprecian los eventos de un calendario compartido que guarda la información de un evento de Circa

5.2.2.2 Integridad y confidencialidad de la información

Aunque se consiguiera guardar los datos de un evento empleando los servicios de Google Calendar, todavía quedaría discutir cómo se debería asegurar la integridad y confidencialidad de éstos. Si, por un casual, un usuario con acceso al calendario compartido de un evento de Circa abriera la aplicación de Google Calendar, podría perfectamente cambiar la información del propio calendario o de los eventos asociados a él. Como este es un problema común al que se pueden enfrentar los programadores que pretendan utilizar los servicios de Google Calendar, la API ya viene provista de mecanismos que aseguren la integridad y confidencialidad de los parámetros almacenados.

Los calendarios tienen una propiedad oculta —no accesible por la interfaz de la aplicación —llamada ***Hidden***, la cual es un booleano que permite, sencillamente, hacer que un calendario no sea visible en la lista del usuario. Esta es la opción más interesante, ya que permite ocultar toda la información de un evento de Circa, haciendo que absolutamente todos sus parámetros sean únicamente modificables a través de los servicios de la API, pero sin perder ninguna de las funcionalidades de Calendar, como las notificaciones.

De esta manera, únicamente quedaría concretar como podría discriminar el sistema los calendarios asociados a Circa de los que no lo están. Se podría emplear un código en alguno de los parámetros del calendario, como la ubicación (*Location*). Esta es una buena opción, ya que es la única propiedad modificable (*writable*) del objeto que no es accesible —al menos, actualmente— por la interfaz de usuario de Google Calendar. Al guardar el evento de Circa en el sistema, se le podría asignar un valor a esta propiedad del calendario asociado, como "Circa_event". Así, cuando la aplicación acceda a los calendarios del usuario, esta sabrá perfectamente cuáles son los que le interesan.

5.2.2.3 Escalabilidad

Como hemos podido observar, la integración de nuestra aplicación en la nube de Calendar parece viable, pero no se ha discutido todavía qué pasaría si se quisiera ampliar sus funcionalidades, y, por lo tanto, nuevos parámetros que guardar. En algún momento, las propiedades del calendario y sus eventos podrían no ser suficientes para cumplir con las necesidades de la aplicación. Por ello, y de forma similar al apartado anterior, Google también ha previsto esta situación, dejando para el uso exclusivo de los desarrolladores externos el objeto *ExtendedProperties* [36]. Esta propiedad de los eventos (clase *Event*) constituye una colección de parejas clave-valor pensadas para almacenar metadatos relacionados con el evento. Pese a que estas parejas no pueden guardar grandes cantidades de información, podrían bastar para guardar pequeñas cadenas de texto, unidades o booleanos, entre otros.

Capítulo 6. ANÁLISIS DE RESULTADOS

En este capítulo se explicarán las pruebas realizadas sobre la aplicación desarrollada y los resultados obtenidos de ellas.

6.1 PRUEBAS

Entre las pruebas realizadas se encuentran:

- Comprobar el correcto funcionamiento visual de todos los elementos de la interfaz según el momento de vida del evento (estatus).
- Revisar si el administrador era capaz de ejercer sus poderes especiales, a la vez que se constataba que los invitados no.
- Verificar que los calendarios, listas y demás elementos se actualizaban a tiempo real conforme a los cambios producidos en el *viewmodel*.
- Constatar que los datos de la interfaz se recogían y el evento era creado satisfactoriamente.
- Comprobar que los datos de los eventos ya creados eran mostrados correctamente por la interfaz, ya fuera en el calendario de la página principal o en la página de gestión de evento.
- Revisar que las fechas propuestas se recogían correctamente y cumplían con las condiciones impuestas, como el número máximo de votos por usuario.
- Verificar que los votos de cada usuario se almacenen correctamente y se tienen en cuenta para el número de votos totales.

Todas estas pruebas se han realizado en un móvil Samsung A50 con sistema operativo Android 9 y en un Samsung S6 con sistema Android 7. También se repitieron compilando como una aplicación la aplicación UWP (Plataforma Universal de Windows) de forma nativa en el ordenador con sistema Windows 10. Aunque los resultados generales fueron satisfactorios respecto a su funcionalidad, al estar pensada como una aplicación con formato

móvil, la interfaz y sus elementos, especialmente los iconos, presentan un aspecto ligeramente deforme.

Pese a que hubiera sido lo ideal, no ha sido posible probar la aplicación en entornos móviles con iOS —específicamente iPhone—. Esto se ha debido a que la compilación del código se debe hacer con Xcode [37], un IDE exclusivo de Apple que no dispone de versión para Windows. Pese a que se realizó un emparejamiento con un iMac de mediados de 2010 [38], el cual permite a Visual Studio compilar aplicaciones iOS desde Windows, el sistema operativo que soportaba era demasiado antiguo como para poder instalar la última versión de Xcode. En consecuencia, la versión de Xamatin.iOS en la que se ha desarrollado Circa no era compilable.

6.2 RESULTADOS

En este proyecto se ha realizado el diseño del sistema y la programación parcial de una aplicación móvil multiplataforma. Los resultados obtenidos durante el desarrollo se pueden dividir en:

6.2.1 INTERFAZ

La interfaz utilizada permite al usuario crear, modificar y visualizar sus eventos de una forma sencilla e intuitiva. Pese a que no permite operaciones esenciales, como compartir los eventos con otros usuarios, su alcance se puede ampliar fácilmente debido a que se trata de un sistema escalable. Además, la capacidad de adaptarse a los diferentes sistemas operativos a través de un único código fuente hace que sea muy sencillo realizar cambios en la misma, ya que no hace falta preocuparse de los controles nativos de cada dispositivo.

6.2.2 GESTIÓN DE DATOS

La estructura de clases desarrollada, al igual que la interfaz, es altamente escalable, lo que garantiza la capacidad de realizar futuras ampliaciones de funcionalidad si fuera necesario.

6.2.3 INTEGRACIÓN CON EL SISTEMA EN LA NUBE

Pese a que no ha sido posible llevarlo a cabo, se han presentado y discutido diversas opciones de inicio de sesión y almacenamiento en la nube, entre las que se encuentran los servicios de Google. Pese a que los modelos propuestos para integrar las clases de Circa con los servicios de la compañía norteamericana pueden no ser completamente factibles en la práctica, se trata de un buen punto de partida para trabajos posteriores.

Capítulo 7. CONCLUSIONES

Este proyecto se ha centrado en conseguir un sistema de gestión de eventos compartidos que innove a otras aplicaciones disponibles en el mercado en varios aspectos. La aplicación desarrollada en este proyecto integra de forma efectiva la interfaz de los calendarios electrónicos con la funcionalidad de los *schedulers*, logrando un diseño intuitivo y sencillo, apoyándose en elementos visuales presentes con los que los usuarios ya se encuentran familiarizados. Esto, junto a su desarrollo multiplataforma tanto para dispositivos móviles (Android e iOS) como ordenadores (Windows 10), hacen que sea accesible para toda clase de usuarios. En cuanto a su funcionalidad, Circa ha conseguido introducir de forma efectiva en el mundo de los *schedulers* la posibilidad de que los invitados puedan proponer sus propias opciones, en lugar del sistema tradicional, el cual ciñe el sistema de voto a las señaladas previamente por administrador. Para ello, se ha desarrollado un modelo de datos acorde a las necesidades básicas de la aplicación.

7.1 OBJETIVOS NO LOGRADOS

Debido a una infravaloración del tiempo requerido para cada apartado, mezclada con las dificultades derivadas de la crisis del COVID-19 en nuestro país, ciertas capacidades prometidas no se han podido llevar a cabo. A pesar de que se ha logrado crear una aplicación que permite crear y gestionar los eventos de forma efectiva, la realidad es que actualmente el sistema desarrollado no permite compartir eventos con otros usuarios. Pese a ello, el estudio realizado sobre los sistemas en la nube indica que es posible no depender de un servidor y base de datos propios, sino que emplear únicamente servicios de terceros, como los de Google, para guardar y gestionar los datos de una aplicación es posible e incluso en algunos casos recomendado.

Por otro lado, originalmente existía la idea de hacer que un evento pudiera ser administrado por varias personas al mismo tiempo, pero finalmente se desechó. Esto fue debido a que no se trataba de una opción muy novedosa, además de que no tendría mucho sentido su

implementación en el estado actual de Circa. Pese a ello, el sistema podría habilitar fácilmente esta opción con ligeras modificaciones, lo que haría posible su inclusión a posteriori si se deseara.

Capítulo 8. TRABAJOS FUTUROS

En este capítulo se enumerarán y se describirán brevemente posibles mejoras de los sistemas existentes o la ampliación funcional de los mismos.

8.1 POSIBLES MEJORAS

- Rediseñar la interfaz para que los usuarios puedan trabajar con fechas y horas en lugar de sólo días completos.
- Mostrar los eventos en el calendario de la página de gestión. El diseño actual no lo permite por limitaciones del *widget*, por lo que podría ser necesario un gran cambio en el *viewmodel* correspondiente.
- Optimizar el uso de memoria y procesamiento de la interfaz, especialmente la de la página de gestión de evento. Rediseñarla si fuera necesario.
- Instaurar mejores sistemas de control para la entrada de parámetros, especialmente para las cadenas de texto.
- Introducir dinamismo en el tamaño de los elementos de la interfaz para que se adapten mejor a las necesidades del usuario, en especial las listas.
- Crear eventos síncronos que mantengan la información mostrada en la página principal actualizada, al menos, cada minuto.
- Seleccionar una gama de colores idónea para la interfaz.

8.2 AMPLIACIONES FUTURAS

- Comprobar el buen funcionamiento de la aplicación en iOS y otros dispositivos Android. Probablemente serán necesarias adaptaciones en código nativo para crear una interfaz realmente adaptable a todos los dispositivos.
- Integrar en la práctica la aplicación móvil con el sistema en la nube. Se recomienda Google debido a que otorga control de inicio de sesión y su API de calendario es de las más completas, además de ser gratuita.

- Crear las páginas de inicio de sesión, gestión de cuentas y de invitación a los eventos. Sin ellas, la aplicación nunca podrá tener una funcionalidad completa.
- Incluir otro tipo de opciones a votar aparte de fechas y horas, como texto. Aunque la aplicación fue concebida desde un primer momento para poder admitir diferentes tipos de opciones, esta ampliación requerirá importantes cambios en la interfaz.
- Permitir crear eventos con múltiples administradores. Esto requeriría ligeros cambios en la política de clases, además de idear e implementar una nueva interfaz específica para ello.

Capítulo 9. RELACIÓN DEL PROYECTO CON LOS ODS

Este capítulo tratará sobre los Objetivos de Desarrollo Sostenible de Naciones Unidas, mejor conocidos por sus siglas ODS, y su relación con el proyecto desarrollado en esta memoria. Toda la información sobre los mismos se ha obtenido de la página web oficial de Naciones Unidas [39].

Figura 57. Objetivo de Desarrollo Sostenible [39]

Los ODS son "un conjunto de objetivos globales para erradicar la pobreza, proteger el planeta y asegurar la prosperidad para todos" establecidos en 2015 por la Organización de Naciones Unidas [39]. Son 17 objetivos, los cuales se concretan a través de 169 metas específicas, cuyo propósito es impulsar el desarrollo sostenible a nivel global. Entre sus aspectos más ambiciosos se encuentran frenar el cambio climático, eliminar el hambre en el mundo e impulsar las energías renovables. Como fecha límite para su revisión, la ONU estableció el año 2030.

Los objetivos se suelen agrupan en tres grandes esferas —social, medioambiental y económica— aunque no quiere decir que los objetivos de estos grupos no tengas correlación: no es suficiente con avanzar en uno si es en detrimento de otro. Por ejemplo, la construcción de una presa hidroeléctrica claramente sería un avance positivo respecto a las energías renovables (objetivo #7) y el trabajo (#8), pero afectaría negativamente a los ecosistemas fluviales (#14) y terrestres (#15). Todos los objetivos deben de avanzar de la mano, sin dejar ninguno demasiado atrás ni demasiado adelante, para alcanzar de esta manera un desarrollo sostenible duradero a nivel global.

9.1 RELACIÓN CON EL PROYECTO

Pese a que pudiera parecer que el proyecto desarrollado no guarde ninguna relación los con los ODS, lo cierto es que si hay algunos con los que comparte ciertos aspectos.

9.1.1 TRABAJO DECENTE Y CRECIMIENTO ECONÓMICO (OBJETIVO #8)

La implantación de la aplicación desarrollada en este TFG supondría una centralización de los recursos a la hora de organizar citas y otros eventos.

En el ámbito laboral, las empresas —especialmente las grandes— tienden a implementar sistemas de eventos sincronizados con agendas y calendarios electrónicos, como Outlook. Pese a que estos sistemas son muy eficientes dentro de la red empresarial, ya que permiten visualizar los espacios libres de todos los empleados a la hora de organizar reuniones, suelen ser inútiles a la hora de entablar encuentros con otras entidades. Como es lógico, las empresas no suelen tener acceso a las agendas de otras organizaciones, lo que hace que organizar eventos sea muy complicado. Debido a esto, se tienden a utilizar otros canales de comunicación, como el correo electrónico o el teléfono móvil, mucho menos eficientes en tiempo que los sistemas internos.

La implementación de la aplicación desarrollada en este TFG podría solucionar este problema, ya que elimina la necesidad de comunicación por vías externas a la hora de organizar cualquier evento. Así, la aplicación centraliza todas las discusiones e

incomodidades que entablan la planificación de encuentros, especialmente si se trata de poner a un gran número de personas en común.

La eficiencia de la entidad se vería afectada positivamente, ya que sus empleados podrían centrarse muchos más en sus obligaciones, sin preocuparse por cómo y cuándo va a tener lugar el encuentro. Al aumentar la eficiencia, mejoraría la productividad de la empresa, lo que impulsaría el crecimiento económico general.

9.1.2 INDUSTRIA, INNOVACIÓN E INFRAESTRUCTURAS (OBJETIVO #9)

Pese a que suele considerarse que las infraestructuras sólo son físicas, como carreteras, edificaciones o redes de cableado, lo cierto es que las infraestructuras de servicios también son de gran importancia en la actividad económica mundial. Dentro de ellas podría ubicarse Circa, ya que, además de introducir innovaciones en el mundo de los *schedulers*, también cubre una necesidad muy común en el ámbito personal y laboral de miles de personas.

Muchas empresas, en especial las pequeñas y las grandes, no pueden permitirse los costos de mantener una infraestructura de comunicaciones. Nuestra aplicación podría servir como un apoyo a estas organizaciones, ofreciéndoles un servicio eficaz y fiable, que, pese a no ofrecer todas las ventajas de una gran empresa como Microsoft o Google, pueda serles de ayuda para organizar citas y encuentros a un precio asequible.

Capítulo 10. BIBLIOGRAFÍA

[1] Microsoft Docs, «What is Xamarin? | .NET», *2020*, 2020. [En línea]. Disponible en: https://dotnet.microsoft.com/learn/xamarin/what-is-xamarin. [Accedido: 22-jun-2020].

[2] Google LLC, «Google Calendar App», 2020. [En línea]. Disponible en: https://play.google.com/store/apps/details?id=com.google.android.calendar. [Accedido: 20-ene-2020].

[3] Google LLC, «Productos | G Suite», *2020*, 2020. [En línea]. Disponible en: https://gsuite.google.com/. [Accedido: 13-jun-2020].

[4] Cabify, «Cabify - Aplicaciones en Google Play», *2020*, 2020. [En línea]. Disponible en: https://play.google.com/store/apps/details?id=com.cabify.rider. [Accedido: 13-jun-2020].

[5] C. S. M. S. SL, «ZITY», *2020*, 2020. [En línea]. Disponible en: https://play.google.com/store/apps/details?id=es.zitycar.carsharing. [Accedido: 06-jul-2020].

[6] Google Developers, «Calendar API | Google Developers», 2020. [En línea]. Disponible en: https://developers.google.com/calendar/. [Accedido: 29-ene-2020].

[7] Microsoft Corporation, «Centro de desarrollo de Microsoft Graph», *2020*, 2020. [En línea]. Disponible en: https://developer.microsoft.com/es-es/graph. [Accedido: 13-jun-2020].

[8] APPS4CITIZENS, «Appgree, una “app” para la participación democrática digital», *elPeriódico*, Barcelona, Spain, 2015.

[9] Podemos, «Podemos», *2020*, 2020. [En línea]. Disponible en: https://podemos.info/.

[Accedido: 09-jul-2020].

[10] Appgree S.A., «Appgree», 2019. [En línea]. Disponible en: https://www.appgree.com/appgree/en/. [Accedido: 06-feb-2020].

[11] Doodle Team, «Doodle App», 2020. [En línea]. Disponible en: https://play.google.com/store/apps/details?id=com.doodle.android. [Accedido: 21-ene-2020].

[12] Tamedia AG, «Doodle to be Acquired by Tamedia», 2014. [En línea]. Disponible en: https://thenextweb.com/insider/2014/01/07/simple-scheduler-app-doodle-acquired-swiss-media-group-tamedia/. [Accedido: 25-ene-2020].

[13] Doodle Team, «Doodle | Premium», 2019. [En línea]. Disponible en: https://doodle.com/premium. [Accedido: 25-ene-2020].

[14] Bash Team, «Bash | Personal event organising - Aplicaciones en Google Play», *2019*, 2019. [En línea]. Disponible en: https://play.google.com/store/apps/details?id=com.vlinderstorm.bash. [Accedido: 29-ene-2020].

[15] Bash Team, «Bash», *2019*, 2019. .

[16] MSPoweruser, «Breaking: Microsoft acquires Xamarin, a leading platform provider for mobile app development», *2016*, 2016. [En línea]. Disponible en: https://mspoweruser.com/breaking-microsoft-acquires-xamarin-a-leading-platform-provider-for-mobile-app-development/. [Accedido: 22-jun-2020].

[17] M. Cuevas, «¿Es Xamarin la mejor tecnología multiplataforma para aplicaciones móviles?», *2019*, 2019. [En línea]. Disponible en: https://www.hiberus.com/crecemos-contigo/xamarin-tecnologia-multiplataforma-aplicaciones-moviles/. [Accedido: 23-jun-2020].

[18] Microsoft Docs, «Información general sobre XAML - WPF», *2019*, 08-ago-2019. [En

línea]. Disponible en: https://docs.microsoft.com/es-es/dotnet/desktop-wpf/fundamentals/xaml. [Accedido: 23-jun-2020].

[19] Microsoft Docs, «Un paseo por C# - Guía de C#», *2020*, 2020. [En línea]. Disponible en: https://docs.microsoft.com/es-es/dotnet/csharp/tour-of-csharp/. [Accedido: 23-jun-2020].

[20] Microsoft Docs, «¿Qué es una aplicación para la Plataforma universal de Windows (UWP)? - UWP applications», *2018*, 2018. [En línea]. Disponible en: https://docs.microsoft.com/es-es/windows/uwp/get-started/universal-application-platform-guide. [Accedido: 23-jun-2020].

[21] Geekstorming!, «Aventuras .NET – Condicionales», *2020*, 28-ene-2020. [En línea]. Disponible en: https://geekstorming.wordpress.com/2020/01/28/aventuras-net-condicionales/. [Accedido: 11-jul-2020].

[22] J. Gossman, «Introduction to Model/View/ViewModel pattern for building WPF apps | Microsoft Docs», *2005*, 2005. [En línea]. Disponible en: https://docs.microsoft.com/es-es/archive/blogs/johngossman/introduction-to-modelviewviewmodel-pattern-for-building-wpf-apps. [Accedido: 22-jun-2020].

[23] M. A. Gómez, «Software Crafters», *2018*, 2018. [En línea]. Disponible en: https://softwarecrafters.io/xamarin/patron-mvvm-xamarin-forms. [Accedido: 24-jun-2020].

[24] Microsoft Corporation, «NuGet Gallery | Home», *2020*, 2020. [En línea]. Disponible en: https://www.nuget.org/. [Accedido: 23-jun-2020].

[25] Xamarin, «NuGet Gallery | Xamarin.Essentials 1.5.3.2», *2020*, 2020. [En línea]. Disponible en: https://www.nuget.org/packages/Xamarin.Essentials/. [Accedido: 23-jun-2020].

[26] Xamarin, «NuGet Gallery | Xamarin.Forms 4.7.0.968», *2020*, 2020. [En línea].

Disponible en: https://www.nuget.org/packages/Xamarin.Forms/. [Accedido: 23-jun-2020].

[27] Syncfusion, «1,600+ Free controls and frameworks for desktop, web, and mobile apps.», *2020*, 2020. [En línea]. Disponible en: https://www.syncfusion.com/products/communitylicense. [Accedido: 23-jun-2020].

[28] Syncfusion, «Overview of Syncfusion Flutter calendar | Scheduler», *2020*, 2020. [En línea]. Disponible en: https://help.syncfusion.com/flutter/calendar/overview. [Accedido: 23-jun-2020].

[29] Syncfusion, «NuGet Gallery | Syncfusion.Xamarin.SfCalendar 18.1.0.59», *2020*, 2020. [En línea]. Disponible en: https://www.nuget.org/packages/Syncfusion.Xamarin.SfCalendar/. [Accedido: 23-jun-2020].

[30] Freepik, «Android app 200 free icons (SVG, EPS, PSD, PNG files)», 2019. [En línea]. Disponible en: https://www.flaticon.com/packs/android-app-9. [Accedido: 10-jul-2020].

[31] Google LLC, «Firebase Authentication | Simple, free multi-platform sign-in», *2020*, 2020. [En línea]. Disponible en: https://firebase.google.com/products/auth?hl=es-419&gclid=CjwKCAjw26H3BRB2EiwAy32zhYLfSx32QYfkJZLZzjggTKMhJQu9HUCam2SPVGIen7H8FVMUClScGhoCXcUQAvD_BwE. [Accedido: 16-jun-2020].

[32] E. Ferreño, «Gmail tiene ya 1.500 millones de usuarios activos», *2018*, 2020. [En línea]. Disponible en: https://www.profesionalreview.com/2018/10/28/gmail-tiene-ya-1-500-millones-de-usuarios-activos/. [Accedido: 16-jun-2020].

[33] Google Developers, «Calendars | Calendar API», *2020*, 2020. [En línea]. Disponible en: https://developers.google.com/calendar/v3/reference/calendars. [Accedido: 17-jun-2020].

[34] Google Developers, «Events | Calendar API», *2020*, 2020. [En línea]. Disponible en: https://developers.google.com/calendar/v3/reference/events. [Accedido: 17-jun-2020].

[35] Google Developers, «Calendar API for .NET», *2020*, 2020. [En línea]. Disponible en: https://developers.google.com/resources/api-libraries/documentation/calendar/v3/csharp/latest/index.html. [Accedido: 18-jun-2020].

[36] Google Developers, «Extended Properties | Calendar API », *2020*, 2020. [En línea]. Disponible en: https://developers.google.com/calendar/extended-properties. [Accedido: 05-jul-2020].

[37] Apple Developer, «Xcode», *2020*, 2020. [En línea]. Disponible en: https://developer.apple.com/xcode/. [Accedido: 10-jul-2020].

[38] A. Support, «iMac (21,5 pulgadas, mediados de 2010) - Especificaciones técnicas», *2010*, 2010. [En línea]. Disponible en: https://support.apple.com/kb/SP588?viewlocale=es_CO&locale=es_CO. [Accedido: 10-jul-2020].

[39] ONU, «Objetivos y metas de desarrollo sostenible – Desarrollo Sostenible», *2020*, 2020. [En línea]. Disponible en: https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/. [Accedido: 06-jul-2020].