Vestigial - Hokuro Inc

Curso 2021/2022

Ingeniería de Sistemas Móviles

Hokuro Incorporation

Hokuro Inc

Vestigial - Hokuro Inc

Curso 2021/2022

Llevado a cabo por

Hokuro Incorporation

Miembro	Correo
Rubén Borrego Canovaca	i82bocar@uco.es
Juan Antonio Fernández Díaz	i82fedij@uco.es
Ángel Fuentes Almoguera	i82fuala@uco.es
Javier García Ibáñez	i82gaibj@uco.es
Pedro Pablo García Pozo	i82gapop@uco.es
Christian Luna Escudero	i82luesc@uco.es

Índice general

1

Introducción

La idea general del proyecto consistiría en una **aplicación móvil** la cual ofrecería un ***organizador de contactos y eventos*** enfocado al ***área de trabajo***. Para ello, debería de combinar las funciones de un **calendario de eventos** con las de una **agenda de contactos** mejorada, permitiendo a los usuarios tener sus propias **listas de tareas** similar a una “To Do List” para así, organizar las tareas pendientes. De esta forma los posibles stakeholders, podrán fácilmente configurar sus propias listas de tareas, grupos de contactos y eventos.

Debido al auge de las nuevas formas de comunicación y al uso estandarizado de la **Tecnología NFC**, la aplicación permitirá el uso de NFC para uso de la transferencia de información. Esta información podría ser desde los datos de un contacto, hasta el conjunto de contactos de un usuario. Incluso se podría compartir un evento mediante de esta forma. Un aspecto a tener en cuenta sería la posibilidad de leer esta información de una tarjeta de visita preparada de antemano.

La **ventaja** de combinar la agenda de contactos y calendario es el poder **asignar eventos a contactos o viceversa de forma muy sencilla e intuitiva**. Además de permitir así que cualquier usuario sepa en todo momento los contactos que participan en el evento o que se encuentran relacionados.

2

Objetivos del Problema

El principal propósito de nuestro sistema es el desarrollo de una aplicación móvil que almacene los contactos del usuario y sirva como gestor de estos y de los eventos programados por el usuario. Para ello, es necesario que la aplicación a desarrollar cumpla con cierto conjunto de objetivos los cuales son de vital importancia a la hora de obtener un resultado eficiente, útil, escalable y seguro o, lo que es lo mismo, un producto de calidad.

- La aplicación móvil tiene que ser eficiente, ya que al ser de uso diario y teniendo en cuenta el poco tiempo que pueden perder los usuarios a los que esta va dirigida, el tiempo de respuesta debe de ser ínfimo para provocar el mínimo retraso y perdida de tiempo posible.
- La aplicación móvil tiene que ser útil, es decir, debería de cumplir con los requisitos especificados de forma que logre satisfacer las necesidades expuestas por nuestros usuarios.
- La aplicación móvil tiene que ser escalable con lo que, aunque aumentemos el tamaño del dominio de la aplicación, esta seguirá funcionando con un desempeño y una eficiencia igual a la que tenía con un dominio más pequeño.
- La aplicación móvil tiene que ser segura ya que se trabajan con datos privados de los usuarios en nuestra base de datos, y es de vital importancia que estos se mantengan de forma privada, además de que cualquier tipo de conexión o login debe de ser seguro también.

Además de todas estas metas, también se debe tener en cuenta el coste del desarrollo de la aplicación, tanto de forma material como intangible (a nivel de tiempo, ideas, etc.), pero en nuestro caso no vamos a tenerlo en cuenta al ser un proyecto de universidad.

3
Antecedentes

3.1. Aplicaciones que se utilizan a partir

Como podemos ver en este caso se pretende agilizar y mejorar la gestión de contactos, por ello la aplicación en primer lugar debería de permitir importar los datos de los contactos que ya posee el usuario en otras aplicaciones y de esta forma poder agregarle a estos “contactos”, los diferentes campos que proporciona nuestra aplicación.

Otro aspecto sería la posibilidad de enviar mensajes/direccionar a otras aplicaciones de mensajería como WhatsApp y/o Telegram, para poder enviar/llamar a nuestros contactos. (Estamos estudiando la evolución de signal como posible alternativa a las anteriormente mencionadas)

3.2. Aplicaciones que se utilizan de base para ofrecer nuestra aplicación

A la hora de encontrar aplicaciones similares a la que pretendemos desarrollar, se nos puede venir rápido a la cabeza la gran suit de aplicaciones de Google que ofrece de forma independiente servicios como:

- **Google Calendar**, sería la aplicación más representativa para la gestión del calendario, ya que como hemos planteado al principio en la **Introducción 1**, podemos ver como uno de nuestros puntos fuertes de la aplicación sería la posibilidad de gestionar nuestros **eventos**, y de esta forma poder vincular los diferentes contactos que se encuentren relacionados con este. Dichos contactos siempre deberán de ser del usuario que vaya a crear el evento.
- **Google Tasks**, en este caso sería la aplicación propuesta por Google, la cual pretende únicamente permitir a los usuarios crear sus propias listas de tareas, las cuales se encuentran vinculadas a su usuario. En nuestro caso nosotros unificaríamos las posibles listas de tareas que el usuario poseyera con los diferentes contactos y eventos del usuario.

Propiamente dicho, sabemos que la idea principal del desarrollo de esta aplicación sería el tema de sustituir la aplicación de contactos de los teléfonos, ya que actualmente la gran mayoría de servicios poseen una baja personalización en cuanto a los posibles campos para cada **contacto**. Por ello, con nuestra aplicación permitiríamos la posibilidad de agregar más campos como por ejemplo **oficio, precios, etc.**. Ya que lamentablemente muchos usuarios poseen problemas a la hora de buscar a sus contactos de la empresa o sus asesores.

3.3. Arquitectura Hardware de las cuales nos basamos

La arquitectura que hemos usado para este proyecto es la arquitectura cliente-servidor.

Básicamente tenemos una **Raspberry Pi 4b** que hace las veces de servidor, teniendo instalados en ella tanto un servidor de aplicaciones **(Apache Tomcat)** como un servidor de base de datos **(MySQL)**. Hemos instalado el servidor de aplicaciones porque, aunque la aplicación que vamos a desarrollar es para móviles, antes de hacer la traducción a APK hemos ido haciendo pruebas en navegador web por lo que nuestros ordenadores han hecho las veces de clientes, haciendo peticiones que han sido servidas por Apache Tomcat, que a su vez hace peticiones al servidor de base de datos de MySQL y que son servidas por este. Al hacer la traducción a APK de la aplicación dejaremos de hacer peticiones al servidor de aplicaciones para hacerlas exclusivamente a MySQL.

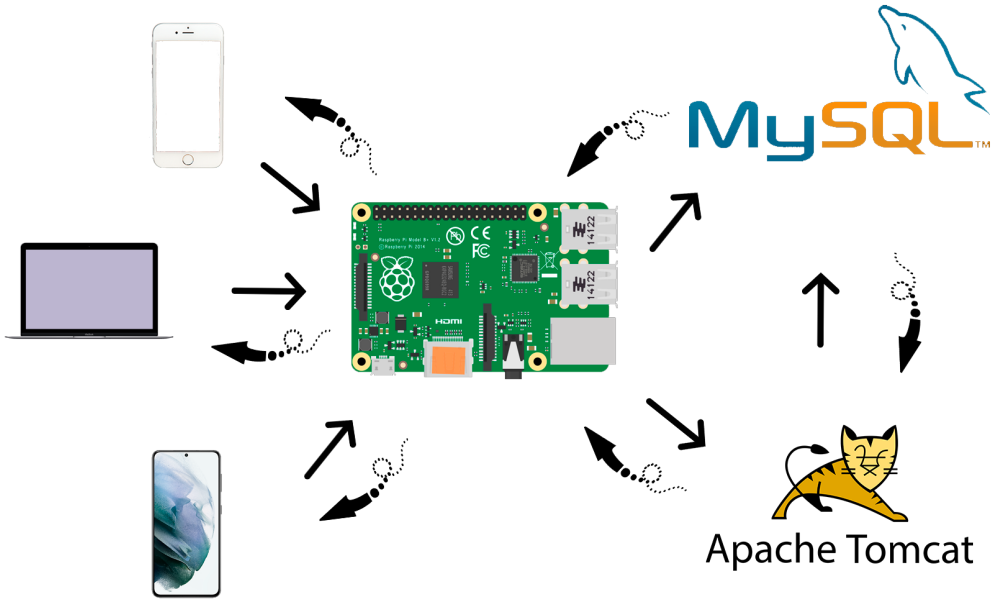


Figura 3.1: Arquitectura Hardware de Vestigial

3.4. Arquitectura Software de las cuales nos basamos

Como ya hemos trabajado anteriormente con el desarrollo de aplicaciones web, decidimos basarnos en la arquitectura Modelo-Vista-Controlador, en la cual en nuestro caso los Modelo y Controladores serían implementados en **Java**, siendo las vistas desarrolladas con la tecnologías de **Ionic** y **Angular**.

En un principio se deberán de realizar diferentes controladores para cada una de las funcionalidades de los componentes que se planteen, en la siguiente **Figura 3.2**, podremos encontrar un ejemplo para nuestro proyecto.

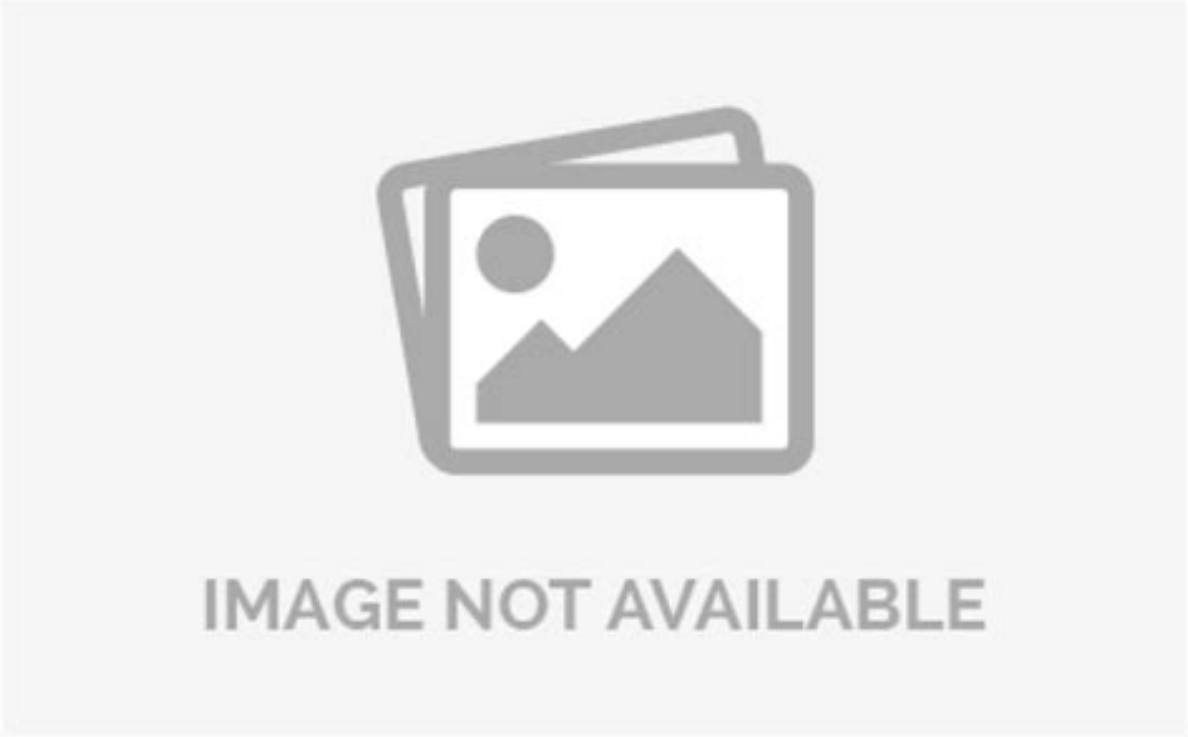


Figura 3.2: Arquitectura Software de Vestigial

4

Recursos

En cuanto a los posibles recursos que cuenta el proyecto a desarrollar, debemos de hacer la distinción en los componentes para el tema de hardware necesario para distribuir la aplicación como los diferentes dispositivos utilizados para las pruebas pertinentes en el desarrollo.

En cuanto al hardware para ofrecer nuestro servicio:

- **Raspberry pi 3B+**: La cual posee un procesador de 1.4Ghz, 1GB de RAM, una conexión de Ethernet a 300 Mbps.
- **Raspberry pi 4B**: posee un procesador de 1,5 GHz, 4 GB de RAM, una conexión de Ethernet a 1Gbps.

Para el tema de las pruebas podemos utilizar los siguientes dispositivos:

- Samsung Galaxy S10
- Xiami Mi 9 Lite
- IPhone 11

5

Análisis y Alcance del problema

5.1. Análisis del problema

Un problema común entre muchas personas es la gestión de su agenda de contactos debido a la gran cantidad de contactos que puede tener almacenados, que además se acentúa cuando se empiezan a repetir los nombres. También se puede dar el caso de que una persona sea muy desorganizada u olvidadiza, por lo que tener un calendario y listas de eventos pueden ayudarle en su día a día.

Nuestra aplicación viene a ayudar a corregir estos problemas de una manera sencilla ya que, aunque no son graves, siempre es de agradecer el poder resolverlos.

5.2. Alcance del problema

El alcance del sistema será permitir al usuario gestionar contactos y eventos a través de la aplicación móvil que vamos a crear. Para ello se hará uso también de la propia tecnología NFC presente en la mayoría de los dispositivos móviles actuales y con la cual ganará bastante en versatilidad. Será posible la sincronización con la nube en el caso de uso con varios dispositivos o en local.

6

Metodología de Trabajo (SCRUM)

6.1. Introducción

A la hora de desarrollar el proyecto, como sabemos debido a la situación del producto a llevar a cabo, nos encontramos en una situación donde una metodología ágil, nos ofrece la mejor forma de trabajar.

Los miembros integrantes para el proyecto actual, contaban con experiencia aplicando el desarrollo de software mediante una metodología ágil, en concreto con la Metodología SCRUM, la cual pretende desarrollar el proyecto mediante un conjunto de iteraciones en las cuales se irán desarrollando partes de la aplicación objetivo. De esta forma en cada iteración (sprints) cada grupo se reunirá para la asignación de tareas y comunicar la situación actual del grupo de trabajo. Como sabemos la metodología SCRUM fomenta los **Daily Meeting**, que consisten en reuniones diarias en las cuales se comentarán tres aspectos fundamentales (¿qué se hizo ayer?, ¿Qué es lo que se va a hacer hoy? y ¿has tenido algún problema durante el desarrollo?) Con esto se pretende que todos los grupos de trabajo se mantengan al tanto de la situación actual. Tras realizar cada Sprint, se deberá de realizar una reunión en la cual se comentará los aspectos negativos y positivos de lo realizado en este Sprint (Sprint Retrospective).

6.2. SCRUM

La realización del Scrum en nuestro proyecto se ha realizado con la ayuda de la aplicación de tablones y metodologías ágiles, Taiga. Con Taiga hemos podido trabajar este Scrum a través de Sprints, cada uno correspondiente al tiempo exacto de una semana.

Primeramente y, antes de comenzar a describir los Sprints, expliquemos un poco la división del equipo **(Figura 6.1)**:

Como se puede apreciar se realizo una división de tres grupos:

- Por un lado encontramos a Christian como Product Owner. Como producto owner ha sido capaz de llevar a cabo su labor de forma incorregible: ha sido capaz de coordinar a todo el equipo, convocar meetings y reuniones tanto semanales como de urgencia, enderezar al equipo de trabajo y generar una correcta división de grupos para un óptimo desarrollo de la aplicación.
- En el apartado de back, encontramos a Pedro Pablo y Ángel, los cuales se encargarán del backend de la aplicación. Esto se refiere realmente al funcionamiento y programación de los servicios así como a la organización de las conexiones y enrutamientos de las funcionalidades. Al ser una tarea tan compleja para dos personas, nos hemos visto en la obligación de aumentar el número de miembros de este grupo en dos más: Christian y Rubén. Ambos han servido de apoyo para este apartado de la aplicación y entre los cuatro han sido capaces de generar el código correspondiente en un tiempo récord contando con el hándicap del conexionamiento con las vistas hechas en Ionic y Angular, un lenguaje de programación al que no están acostumbrados.
- En el apartado de front posicionamos a dos personas: Javier y Juan Antonio, quienes se encargarán de aprender y manejar ionic y angular para poder generar las vistas. Visto el rendimiento del grupo, hemos creído conveniente el cambio de Juan Antonio al apartado de la documentación mientras que Ángel le ha sustituido en el desarrollo de vistas.
- Por último, Rubén se ha encargado del apartado de diseño. No solo cabe destacar su labor con respecto a como se ve la aplicación y las diferentes funcionalidades sino también a los variados

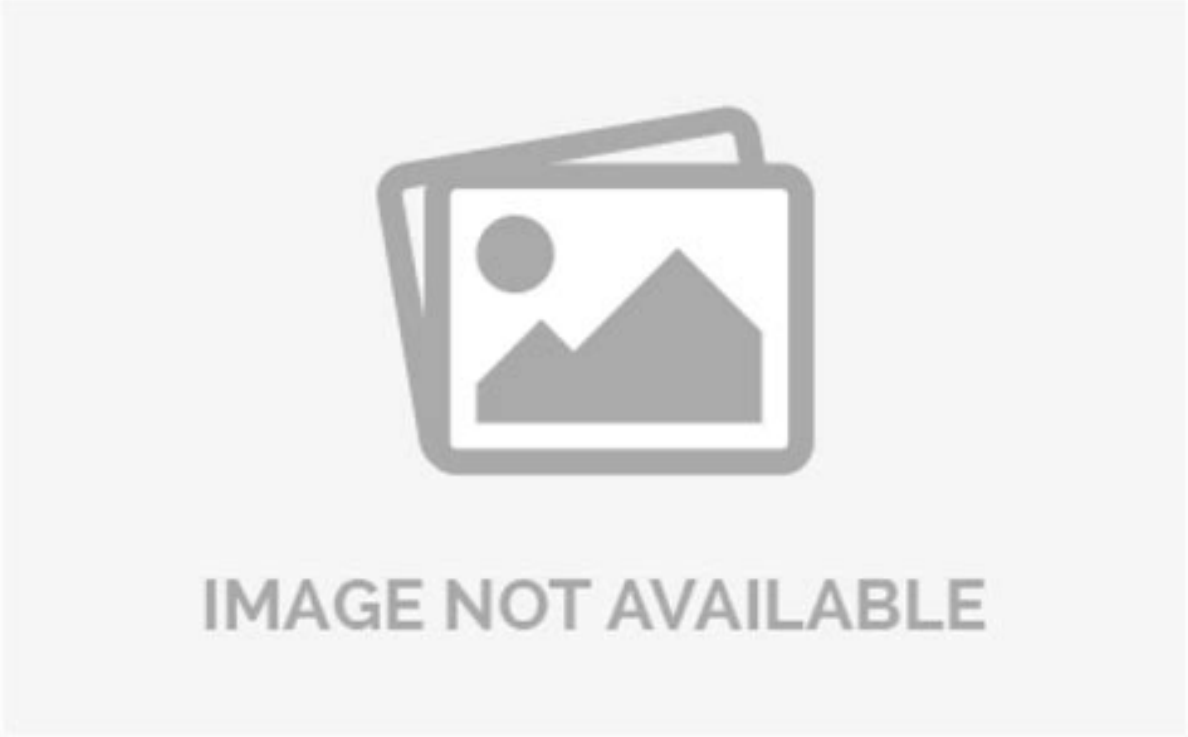


Figura 6.1: IMAGEN DE LA SEPARACIÓN DE GRUPOS DE TRABAJO

aspectos gráficos en temas de animación, scroll, cambio de funcionalidad o visualización general más sobria y sencilla.

Además, a lo largo de todo el proyecto, Juan Antonio se ha encargado de dejar por escrito en las llamadas Wikis el desarrollo semanal de la aplicación junto a los objetivos cumplidos y problemas planteados en cada Sprint.

6.2.1. Desarrollo Metodología SCRUM

Para comenzar a explicar como se han planteado los sprints, cabe destacar el uso de los epics y etiquetas para facilitar la clasificación y vista previa de las diferentes tareas.

Hemos creído conveniente la división de este proyecto en 7 Sprints, cada uno con alrededor de 1 semana de duración y con 3 reuniones semanales: lunes, jueves y domingo. En cada Sprint hemos abordado diferentes temas en conjuntos:

En el primer Sprint, nos centramos en la elicitación de requisitos y en la creación de diagramas y casos de uso para poder tener una idea mucho más clara de las necesidades del usuario. También comenzamos con la creación y gestión de una base de datos rígida, eficiente, integra y segura y los prototipados y mock ups de la interfaz de usuario.

A partir del segundo Sprint, comenzamos con la programación del proyecto, primeramente con los DAO, haciendo también la configuración del servidor y los DTO, además también nos encargamos de generar unas buenas especificaciones de procesos de negocio.

Durante el tercer Sprint terminamos con los DAO refactorizándolos y sus pruebas y, mientras Javier y Ángel comenzaban con el aprendizaje de angular e ionic, Christian y Pedro realizaban alguna funcionalidad como iniciar o cerrar sesión.

Durante el cuarto y quinto Sprint se desarrollo el grueso de la aplicación: funcionalidades y vistas. Cada grupo de trabajo trabajaba con sus respectivos miembros para ir poco a poco programando tanto las vistas como los diferentes servicios de la aplicación.

Ya en el sexto Sprint, aun con una serie de problemas que se explicarán más adelante cuando se muestren las wikis, se pudo finalmente conexionar las diferentes facetas de la aplicación. Se consiguió

un enrutamiento optimo y sencillo que pudiese tanto enlazar vistas con controladores como los propios controladores entre si además de conectar con la base de datos y el servidor, creados ambos en el primer sprint. Además, se llevaron a cabo diferentes pruebas para poder asegurar que cada servicio hacía aquello para lo que se había programado.

Finalmente, en el último Sprint, el séptimo, llevamos a cabo una emulación y posterior prueba y descarga de la aplicación en un smartphone para poder asegurar el correcto funcionamiento conexionado y enrutamiento de todos los servicios.

Durante cada uno de los Sprint, cada uno tuvo su función y, aunque unos programasen y otros estuvieran con la documentación se llevo a cabo una labor impecable en un tiempo ínfimo.

Otras metodologías

Además de no solo usar la metodología Scrum basada en Sprints de la que disponíamos gracias a Taiga. Taiga también nos ofrece el uso de Kanban y, de esta forma y al unísono, pudimos llevar tanto los Sprint como el Kanban para poder comparar sus puntos débiles y fuertes. Así pues, a la par que con el movimiento de tarjetas en el Sprint de Taiga, se movía la tarjeta del tablón del Kanban.

En cuanto a los Epics, decidimos primeramente generar Epics para cada uno de los Sprints y de esta forma no confundir las tareas y poder asociarlas a sus diferentes sprints. Además, también decidimos posteriormente, crear epics para cada uno de los componentes y saber de forma mucho más rápida y sencilla que tarea esta relacionada con que componente y de que tareas esta compuesto el componente.

Aunque sí los usamos, el uso de los Issues no fue muy notorio en nuestro Scrum, simplemente lo usamos dos veces para poder ponernos de acuerdo en decisiones como el nombre del proyecto o en el uso de estructuras de datos en diferentes dao y dto. Para finalizar el uso de wikis estuvo completamente a cargo de Juan Antonio Fernández, quien actualizaba las páginas para poder generar un diario que ayudase a llevar las diferentes actividades, objetivos y dificultades que se nos plantearon en cada semana. Este diario se puede encontrar tanto en el proyecto de **Taiga**, como en la siguiente **sección**.

6.2.2. Diario del Proyecto

En este apartado del trabajo expondremos las diferentes acciones de cada uno de los miembros del equipo durante las distintas semanas de trabajo y los errores y problemas encontrados junto con las diferentes soluciones y elecciones que tomamos.

Semana 1:

Durante esta **semana**, el grupo de trabajo se centró en fijar el tema del proyecto, especificando funcionalidades, y restricciones que debe seguir nuestra aplicación final.

Estuvimos barajando diferentes posibilidades de temática como es el caso del famoso juego Pokemon, para generar una Pokedex (una base de datos con cada uno de los Pokemon existentes junto con toda su información relacionada). No obstante, esta idea no terminó de convencernos y, siguiendo las indicaciones del profesor, decidimos hacer una agenda de contactos que también permitiera administrar un calendario con eventos.

Tras tener un par de reuniones entre los miembros del grupo decidimos que el trabajo seria sobre la agenda anteriormente mencionada.

Para poder optimizar al máximo el rendimiento del equipo, decidimos dividir las tareas principales entre los miembros del equipo:

- Javier se centro en el apartado del diseño y mock-ups.
- Pedro Pablo se ha encargado del servidor y la base de datos.
- Ángel, Rubén y Juan Antonio nos encargamos de la extracción y especificación de requisitos y la especificación y diagramas de casos de uso.
- Por último, Christian se ha encargado de ayudar de forma general a todos los grupos, coordinándonos y ofreciendo asistencia a cualquier grupo de los ya creados que lo necesite.

Semana 2:
Durante esta **semana** hemos seguido trabajando según los grupos que se nos asignaron en la anterior:

- Javier ha terminado bastantes aspectos del **diseño y previsualización de la página web** con la ayuda de Christian. Esto nos favorece mucho a la hora de la programación, para poder tener en cuenta ***como se debería de ver*** la aplicación y ***como debería funcionar*** además de los posibles enlaces entre las pestañas y funcionalidades que tienen.
- Pedro Pablo y Christian han conseguido generar una **base de datos** segura y bien relacionada para nuestro proyecto. Esto ayudará al almacenamiento de datos de nuestra aplicación. Además, la base de datos esta perfectamente acotada en el sentido del uso de restricciones como **constraints y triggers**, lo que va a permitir asegurar la ***integridad, seguridad, eficiencia y robustez*** de esta.
- Por último, Ángel y Juan Antonio nos hemos encargado de ***generar lo casos de uso y especificarlos*** mientras Rubén se encargaba de generar algunos diagramas y gráficas. La **especificación de casos de uso** se ha hecho basándonos en los requisitos para poder vislumbrar un poco como sería el funcionamiento de la aplicación. Los diagramas, por otro lado, se han realizado de la forma más precisa posible en base a estos casos de uso.

Semana 3:
Durante esta **semana** hemos seguido trabajando según los grupos que se nos asignaron en la anterior:

- Javier se ha familiarizado con el uso de Ionic y Angular y ha adelantado el trabajo de las vistas que se le han asignado. Ionic y Angular nos ayudarán posteriormente con el desarrollo y mejora de la previsualización de la aplicación.
- Pedro Pablo se ha encargado de la refactorización de los DAO. Esto mejora exponencialmente la eficacia y el entendimiento de estos además de que simplifica bastante la programación y comentario de los mismos.
- Christian se ha encargado de la programación de los test con ayuda de Ángel. La creación de estos test facilitará la labor de programación y, sobre todo la depuración y corrección de errores en el código, lo cual aumentará de forma considerable la velocidad de trabajo.
- Rubén ha finalizado los diagramas que tenía pendientes de la anterior semana y ha comenzado a programar algunas vistas de la aplicación. Con la ayuda de Javier y de Juan Antonio, estas vistas serán correctamente depuradas y testeadas junto al resto del código subido.
- Por último, Juan Antonio se ha encargado de la visualización del Taiga y revisión de la documentación. Esto incluye mejora visual del tablero, implementación Kanban paralela a la implementación Scrum, uso de etiquetas y Epics para una mejor clasificación de las tareas y la descripción de las diferentes historias de usuario.

Semana 4:
Durante esta **semana** hemos seguido trabajando según los grupos que se nos asignaron en la anterior:

- Javier ha adelantado una gran porción de las vistas y a él se ha unido Pedro Pablo, quien ha comenzado a familiarizarse con Ionic para poder brindarle algo de ayuda a Javier con las vistas.
- Rubén y Ángel están encargados del desarrollo de Beans y la parte del código restante relacionada con el backend. Aunque es una ardua tarea, Christian se ha encargado de gran parte de los controladores y puede apoyarles para depurar y probar las diferentes partes del backend sobrantes. Entre los tres, han sido capaces de avanzar una gran parte de la programación de la aplicación esta semana
- Por último, Juan Antonio se ha encargado de la visualización del Taiga y revisión de la documentación. Esto incluye mejora visual del tablero, implementación Kanban paralela a la implementación Scrum, uso de etiquetas y Epics para una mejor clasificación de las tareas y la descripción de las diferentes historias de usuario.

Semana 5:
Durante esta **semana** hemos trabajado en dos grupos para finalizar con la aplicación y dejarla a punto y preparada para la exposición en clase:

- Esta semana, como se ha expuesto anteriormente, hemos trabajado en dos grupos para poner a punto la aplicación. El primer grupos está compuesto por Christian y Ángel. Ellos se han encargado de un problema que nos ha surgido a la hora de enlazar backend y frontend. Debido al uso de tecnologías y métodos de programación tan recientes como Ionic y Angular, el enlace de las dos caras de la aplicación se ha dificultado bastante. Hemos intentando enlazar mediante enrutamiento usando mapeados o diferentes ficheros para poder enlazarlos pero aún continúan averiguando como realizarlo. Como esta tarea es bastante complicada y es necesario que sepan las salidas del frontend, el resto del grupo estamos intentando ayudar todo lo posible para que este enlace sea posible cuanto antes.
- Rubén, Javier y Pedro Pablo han continuado con las vistas restantes para poder dejarlas listas y preparadas para probarlas una vez se realice el enlace con el backend.
- Por último, Juan Antonio se ha encargado de la visualización del Taiga y revisión de la documentación. Esto incluye mejora visual del tablero, implementación Kanban paralela a la implementación Scrum, uso de etiquetas y Epics para una mejor clasificación de las tareas y la descripción de las diferentes historias de usuario.

Semana 6:
Esta **semana** ha sido probablemente de las más duras que ha tenido el equipo de trabajo orientado a la programación de la aplicación:

- Durante esta semana, tanto backend (Christian y Pedro Pablo) como frontend (Javier y Ángel) han trabajado en conjunto para poder conexionar ambas partes. Durante 1 semana completa se ha estado intentando conexionar ambas partes sin éxito debido a la nueva tecnología usada Ionic la cual nuestro equipo de trabajo a tenido que aprender en un tiempo récord. Finalmente y, tras 6 días de duro trabajo sin descanso para poder realizar este enlace y enrutamiento, ha sido posible realizarlo, dejando la aplicación un paso más cerca de dejarla a punto.
- Por otro lado, Juan Antonio se ha encargado de la visualización del Taiga y revisión de la documentación. Creación de tareas y asignación de estas a los miembros y watchers. Se ha revisado también la documentación corrigiendo faltas o realzando la lectura del documento para que esta sea mucho más sobria, fácil y comprensible.

Semana 7:
Esta **semana** ha sido algo más tranquila que la anterior. Al ser la recta final del proyecto, nos hemos esforzado todo lo posible en esta última semana para afinar los puntos que necesitaban de revisión:

- Durante esta semana todos los miembros del equipo han trabajado juntos para dejar preparada la aplicación. Hemos probado todos los servicios disponibles y asegurado su correcto funcionamiento y nos hemos informado a cerca de Android Studio, para poder emular la aplicación en un móvil real y testear correctamente los diferentes flujos y conexiones. Pedro, Ángel, Christian, Javier y Rubén se han encargado de esta última parte que compone desde y emulación de la aplicación. Además, también han sido capaces de traspasar la aplicación a un dispositivo móvil para poder testear su funcionamiento en uno.
- Por otro lado, Juan Antonio se ha encargado de la visualización del Taiga y revisión de la documentación. Creación de tareas y asignación de estas a los miembros y watchers. Se ha revisado también la documentación corrigiendo faltas o realzando la lectura del documento para que esta sea mucho más sobria, fácil y comprensible. Además, ha desarrollado un texto sobre Scrum que se añadirá más adelante a la documentación.

Tras ver este pequeño diario de trabajo podemos ver el gráfico de quemados del proyecto entero **(6.2)**, en el cual podemos ver como (CONCLUSIONES FINALES DE SCRUM).

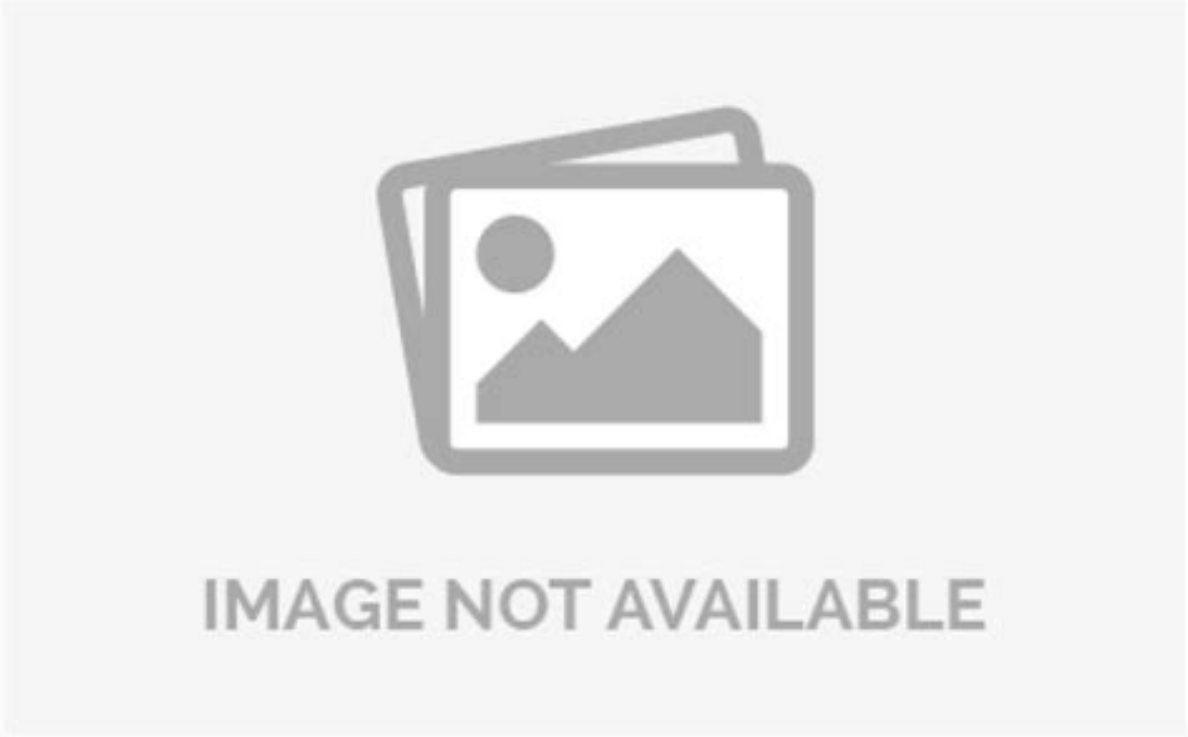


Figura 6.2: IMAGEN GRAFICO DE QUEMADOS

7

Captura de Requisitos

A la hora de realizar la captura de requisitos, los integrantes del grupo fueron reunidos dirigidos por el Scrum Master, el cual presento la idea base y con ellos se llevó a cabo una Brainstorm con la cual se pretendía obtener una información básica para presentar el proyecto. Como podremos ver más adelante una vez realizada la **Brainstorm**, tuvimos una breve entrevista con uno de los stakeholder el cual posteriormente podremos ver las diferentes respuestas que ofreció.

7.1. Brainstorm

A la hora de llevar a cabo esta técnica de extracción de requisitos, los diferentes miembros del aportaron sus experiencias en proyectos similares anteriores o incluso sus inseguridades sobre el problema a resolver. Esta reunión originó diferentes temas que se le iban a presentar a nuestro cliente en la entrevista que se tenía planeada, algunos de estos temas eran:

- Como se deseaba la gestión de contactos (Características deseadas para cada uno)
- Presentación de la idea, y el cómo se trabajaría para conseguir los objetivos planteados
- Posibles métodos de encriptado, como por ejemplo MD5, SHA, RSA, más algunas variantes propias, para mejorar la seguridad de los usuarios.
- Limitaciones para los usuarios.
- Preguntas sobre los aspectos estéticos de la aplicación.
- Además de si se deseaba alguna funcionalidad más de las pensadas en un principio.

Ya que los integrantes de este equipo ya habían colaborado en otros proyectos, plantearon la BrainStorm de la siguiente manera **Figura 7.1**

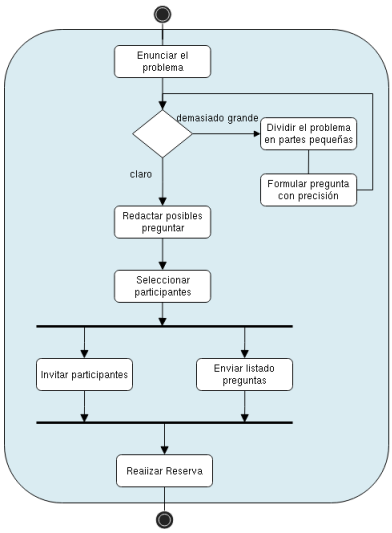


Figura 7.1: Diagrama de actividad realizado en la BrainStorm

7.2. Entrevista

Tras la Brainstorm se decidió realizar una breve entrevista a algunos posibles usuarios de la aplicación de manera que pudieras tener más información de aspectos que se hubieran obviado con anterioridad, de esta forma sabríamos que es lo que desean los usuarios de primera mano.

En cuanto a la entrevista con el cliente, le realizamos diferentes preguntas con el fin de descubrir los posibles usuarios que pueden utilizar el producto, que funcionalidades son claves para el sistema, y si el sistema debería poder ser compatible con otras herramientas software ya existentes o futuras.

A la hora de realizar esta entrevista, se le plantearon las siguientes preguntas:

Entrevistador ¿Se desearía que cada usuario pudiera tener más de una lista de tareas disponible?

Cliente Si, por lo menos que pueda crear una lista con un nombre de categoría y tener varias según las asignaturas o mis proyectos.

Entrevistador ¿Qué campos deben de incluir los contactos de la agenda?

Cliente Pues estaría bien, que pudiera agregar, nombre, teléfono de contacto, aunque que permitiera tener uno o más para cada contacto, ya que tengo contactos que tienen teléfono de empresa y personal, además de que permita correo, ubicación, que me permita escribir una breve descripción etc.

Entrevistador En cuanto el calendario que desea le gustaría poder visualizar en el cualquier tarea o evento que sea próximo a la fecha actual o preferiría que solo mostrar si posee algún evento que sea dicho día y para ver que eventos tener que acceder al día.

Entrevistador A la hora de iniciar en la aplicación le gustaría poder importar los contactos que ya posee en su teléfono de alguna forma.

Cliente Si estaría bien que la primera vez que entre me pregunte si deseo agregar los contactos ya existentes. Como si cambio de teléfono con que inicie en mi cuenta que tenga todos los contactos que tenía en el teléfono anterior.

Tras la entrevista con el cliente, el SCRUM Master decidió dividir el trabajo en dos grupos en los cuales los miembros integrantes serían los encargados de dirigir cada uno de ellos y de mantener informado a tanto al otro encargado como al SCRUM Master, de esta forma se podría estar informado de los aspectos más importantes del proyecto, como estar al tanto del ritmo de trabajo.

Además, se planteó, que cada 4 días se llevaría a cabo una reunión de ambos grupos, adicionalmente de las Daily Meeting para tratar temas de posibles bugs, correcciones, limitaciones o comentarios sobre lo desarrollado en el sprint actual. Esto estaría pensado debido a que uno de los miembros de un grupo sería el encargado de desarrollar los test y las pruebas para los componentes desarrollados.

8

Especificación de Requisitos/ Definición de Requisitos

A la hora de llevar a cabo esta definición de requisitos hemos visto oportuno realizar la clasificación de los estos basándonos en algunos de los estándares con los que el equipo de diseño ha trabajado en proyectos pasados, como por ejemplo la propuesta por Madeja (apoyada por la Junta de Andalucia) o la del IEEE 830.0

De esta forma la clasificación sería, requisitos funcionales, no funcionales y de información. Posteriormente estos requisitos serán pasados al Product Backlog que se llevará a cabo en el desarrollo SCRUM.

8.1. Extracción de Requisitos

8.1.1. Requisitos Funcionales

RF1 El sistema debería de permitir iniciar sesión en la aplicación.

RF2 El sistema debería de permitir iniciar sesión en la aplicación.

RF3 El sistema debería permitir el registro para nuevos usuarios.

RF4 La aplicación debería permitir ver los contactos del usuario.

RF5 La aplicación debería permitir ver la información de un contacto.

RF5 La aplicación debería permitir importar contactos de otras aplicaciones

RF6 La aplicación debería permitir eliminar contactos del usuario

RF7 La aplicación debería permitir añadir un contacto

RF8 La aplicación debería permitir exportar contactos a otras aplicaciones

RF9 La aplicación debería permitir actualizar contactos del usuario

RF10 La aplicación debería de permitir enviar contactos a través de otras aplicaciones de mensajería (WhatsApp, Telegram, etc.)

RF11 La aplicación debería permitir buscar un contacto.

RF12 La aplicación debería permitir ver los eventos del calendario

RF13 La aplicación debería permitir añadir eventos al calendario

RF14 La aplicación debería permitir eliminar eventos del calendario

RF15 La aplicación debería permitir modificar eventos del calendario

RF16 La aplicación debería permitir ver una lista de tareas

RF17 La aplicación debería permitir crear una lista de tareas

RF18 La aplicación debería permitir modificar una lista de tareas

RF19 La aplicación debería permitir borrar una lista de tareas

RF20 La aplicación debería permitir ver un bloc de notas

RF21 La aplicación debería permitir modificar el bloc de notas

RF22 La aplicación debería permitir poder llamar a un contacto desde la agenda.

8.1.2. Requisitos No Funcionales

RNF1 La aplicación debería de ser utilizada en dispositivos móviles

RNF2 La aplicación debería de hacer uso del protocolo NFC/Bluetooth

RNF3 La aplicación debería de ser fácil de usar

RNF4 La aplicación debería de tener una interfaz intuitiva

RNF5 La aplicación debería de usar un formato estandarizado para guardar la información (JSON).

RNF6 La aplicación debería de ser accesible para personas discapacitadas

RNF7 La aplicación debería cumplir con los estándares 27K

RNF8 La aplicación debería permitir notificar los eventos del calendario

8.1.3. Requisitos de Información

RI1 La aplicación debería almacenar la información correspondiente a un contacto (nombre, apellidos, tlf., [expandir])

RI2 La aplicación debería almacenar la información correspondiente a la configuración de la aplicación (tema, idioma, [expandir])

RI3 La aplicación debería almacenar los eventos asignados para día

RI4 La aplicación debería almacenar la lista de tareas del usuario

8.2. Listado Casos de Uso

En la siguiente tabla podemos encontrar los diferentes casos de uso que hemos desarrollado:

Caso de Uso	ID	Descripción
Iniciar sesión	1	El objetivo del caso de uso es permitir a un usuario iniciar sesión en la aplicación.
Cerrar sesión	2	El objetivo del caso de uso es permitir a un usuario cerrar sesión en la aplicación.
Registrarse	3	El objetivo del caso de uso es permitir a un usuario que no está en la base de datos registrarse en la aplicación.
Ver contactos	4	El objetivo del caso de uso es permitir a un usuario ver todos los contactos que tenga guardados.
Ver información del contacto	5	El objetivo del caso de uso es permitir a un usuario ver la información guardada de un contacto.
Buscar contacto	6	El objetivo del caso de uso es permitir a un usuario buscar un contacto de entre los que tiene guardados.
Añadir contacto	7	El objetivo del caso de uso es permitir a un usuario añadir un nuevo contacto.
Eliminar contacto	8	El objetivo del caso de uso es permitir a un usuario eliminar un contacto.
Actualizar contacto	9	El objetivo del caso de uso es permitir a un usuario actualizar los datos guardados de un contacto.
Importar contactos	10	El objetivo del caso de uso es permitir a un usuario importar contactos desde otras aplicaciones.
Exportar Contacto	11	El objetivo del caso de uso es permitir comprimir un contacto en un fichero de extensión compatible para traspasarlo a un dispositivo diferente.
Exportar Contacto Avanzado	12	El objetivo del caso de uso es permitir comprimir todos los contactos en un fichero de extensión compatible para traspasarlo a un dispositivo diferente.
Visualizar calendario de eventos	13	El objetivo del caso de uso es permitir a un usuario visualizar su calendario de eventos.
Añadir evento	14	El objetivo del caso de uso es permitir a un usuario añadir un evento a su calendario.
Eliminar evento	15	El objetivo del caso de uso es permitir a un usuario eliminar un evento de su calendario.
Actualizar evento	16	El objetivo del caso de uso es permitir a un usuario modificar la información de un evento.
Visualizar lista de tareas	17	El objetivo del caso de uso es permitir a un usuario ver su lista de tareas por hacer.
Crear lista de tareas	18	El objetivo del caso de uso es permitir a un usuario crear una nueva lista de tareas.
Borrar lista de tareas	19	El objetivo del caso de uso es permitir a un usuario borrar una lista de tareas ya creada.
Modificar lista de tareas	20	El objetivo del caso de uso es permitir a un usuario modificar las tareas de una lista de tareas.
Ver bloc de notas	21	El objetivo del caso de uso es permitir a un usuario ver su bloc de notas .
Modificar bloc de notas	22	El objetivo del caso de uso es permitir a un usuario modificar su bloc de notas.
Llamar a un contacto	23	El objetivo del caso de uso es permitir a un usuario llamar a uno de sus contactos de la agenda.

Cuadro 8.1: Listado de los Casos de Uso

8.2.1. Modelo de Concepto de Negocio (BCM)

A la hora de realizar el BCM, debemos de tener en cuenta cuales son los conceptos más importantes en el **Dominio Del Problema** a tratar, como podemos pensar en nuestro caso existirán diferentes aspectos los cuales podrían ser denominados con nombres “diferentes”, los cuales hagan referencia a la misma cosa, por ende, tras realizar un estudio tanto en el dominio del problema, como revisando la entrevista y estudiar los requisitos propuestos se han llegado a los siguientes conceptos:

- Usuario
- Grupo de Contactos
- Contacto
- To Do List (Listas de Tareas)
- Tarea
- Evento

Una vez destacados algunos de los conceptos, en la **8.1** podremos ver el diagrama de clases con el que hemos representado gráficamente el BCM.

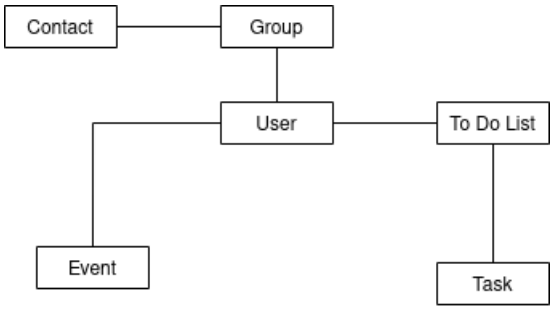


Figura 8.1: Diagrama del BCM (Business Concept Model)

8.2.2. Modelos de Procesos de Negocio

Para realizar el Modelado de los procesos de negocios en primer lugar deberemos de identificar cada uno de los procesos de negocio que nos encontramos en el problema. El listado sería:

- Gestión de la Agenda:
  - Mostrar listado de Contactos.
  - Ver información del contacto.
  - Buscar un contacto.
  - Añadir contacto.
  - Eliminar contacto.
  - Actualizar contacto.
  - Actualizar contacto.
  - Importar Contactos.
  - Exportar Contacto.
  - Llamar a un contacto.
- Gestión del Calendario
  - Mostrar listado de Eventos.
  - Añadir evento.
  - Eliminar evento.
  - Actualizar evento.
- Gestión del To Do List
  - Mostrar lista de tareas.
  - Crear Lista de tareas.
  - Eliminar lista de tareas.

- Modificar lista de tareas.

- Gestión del Bloc de Notas
  - Visualizar Bloc de notas.
  - Modificar Bloc de notas.

Proceso de Negocio Gestión del Calendario

En este proceso de negocio el cliente tendrá acceso al menú de opciones en el cual podrá visualizar su calendario de eventos, como agregar un nuevo evento, eliminar o actualizar un evento. El diagrama se podrá ver en la **Figura 8.2**

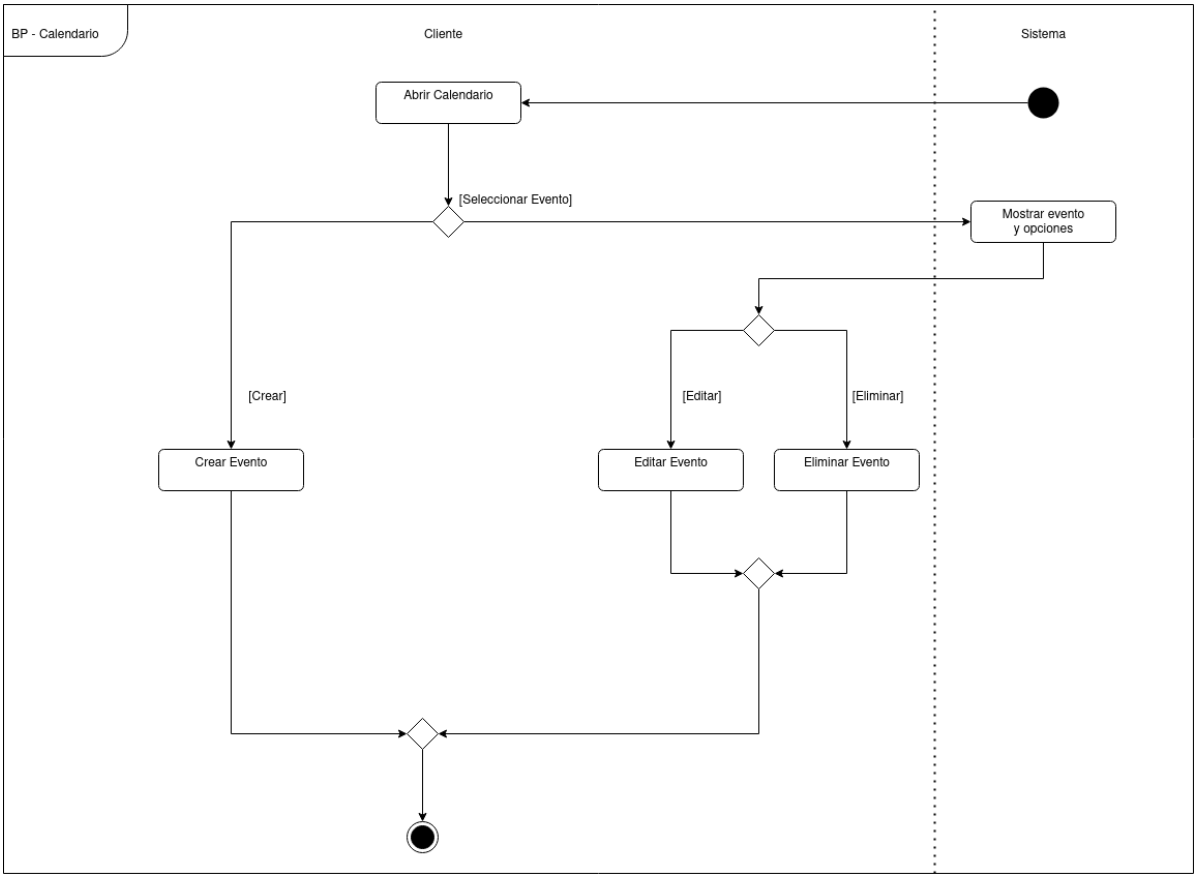


Figura 8.2: Diagrama de Actividad Gestión del Calendario

Proceso de Negocio Gestión del To Do List

En este proceso de negocio el cliente tendrá acceso al menú de opciones en el cual podrá visualizar sus listas de tareas, permitiendo le tanto crear nuevas listas, como agregar una nueva tarea a una lista, eliminar o actualizar una tarea. El diagrama se podrá ver en la **Figura 8.3**

Proceso de Negocio Gestión de la Agenda

En este proceso de negocio el cliente tendrá acceso al menú de opciones en el cual podrá visualizar su listado de contactos, agregar un nuevo contacto, editar su información, eliminarlo. El diagrama se podrá ver en la **Figura 8.4**

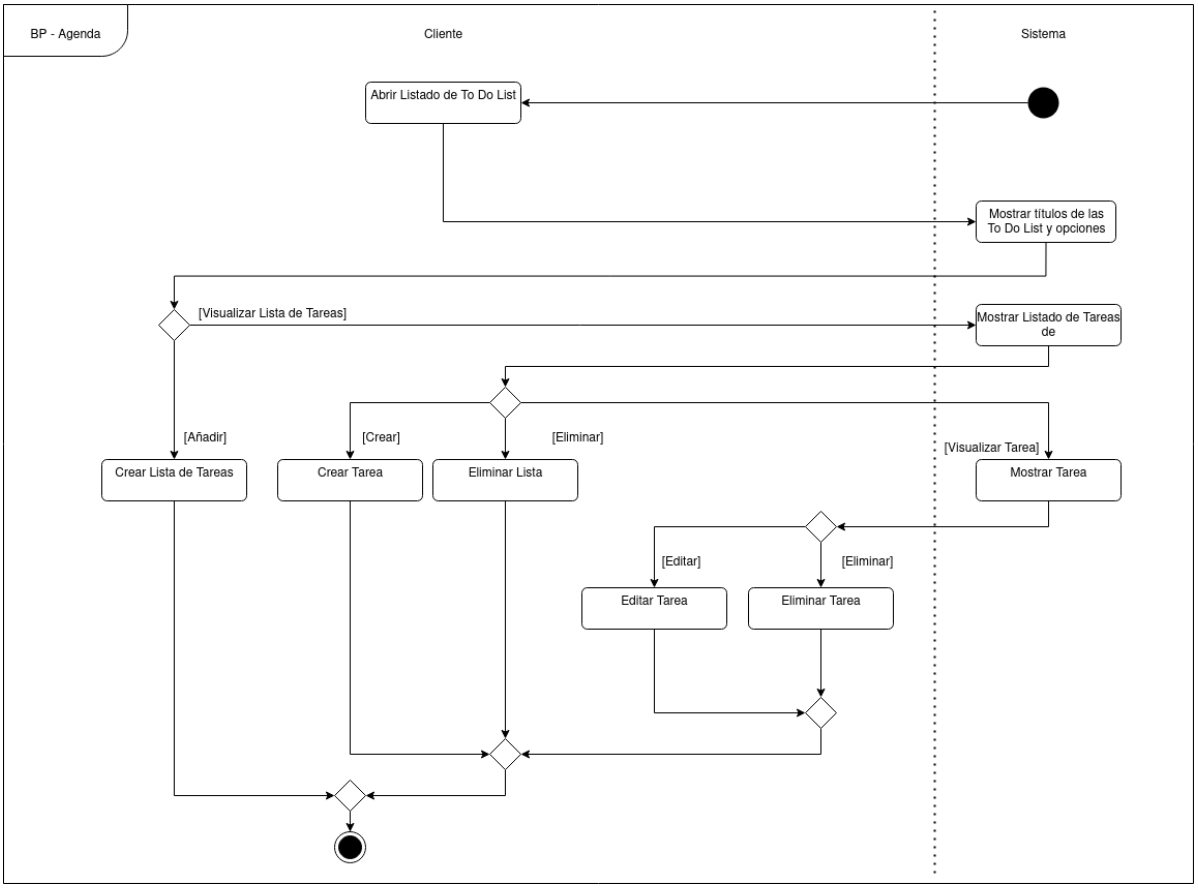


Figura 8.3: Diagrama de Actividad Gestión del To Do List

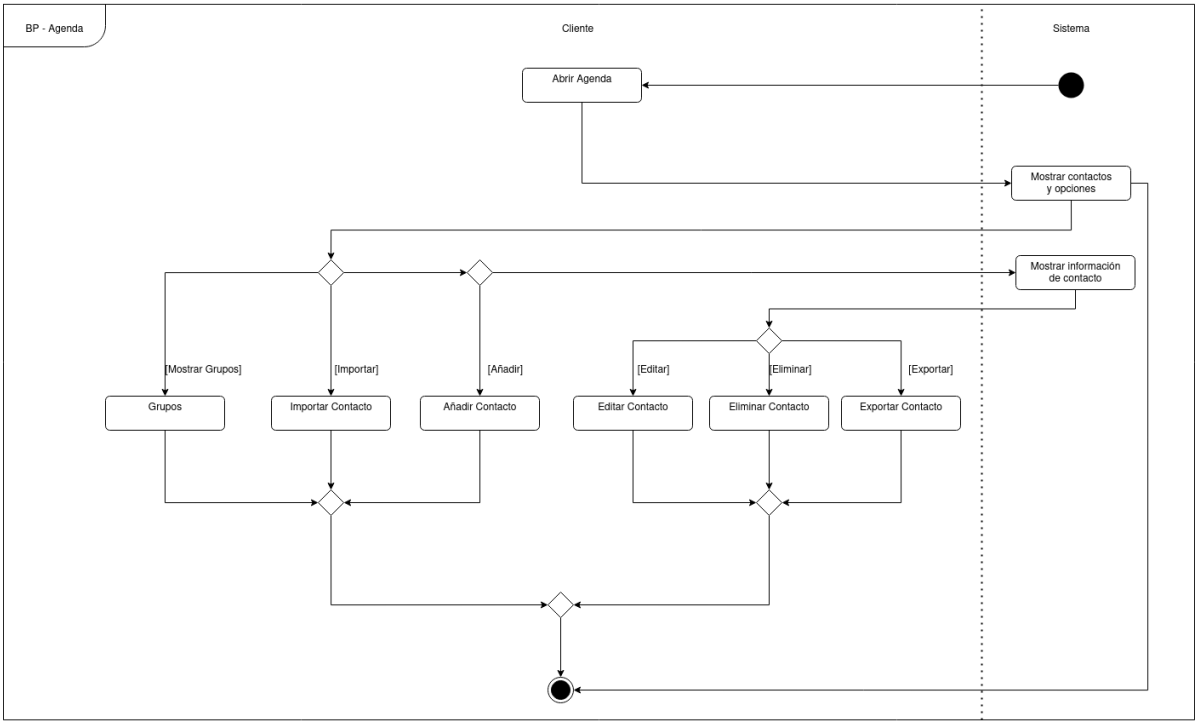


Figura 8.4: Diagrama de Actividad Gestión de la Agenda

Proceso de Negocio Bloc de Notas

En este proceso de negocio el cliente tendrá acceso al menú de opciones en el cual podrá visualizar su listado de contactos, agregar un nuevo contacto, editar su información, eliminarlo. El diagrama se podrá ver en la **Figura 8.5**

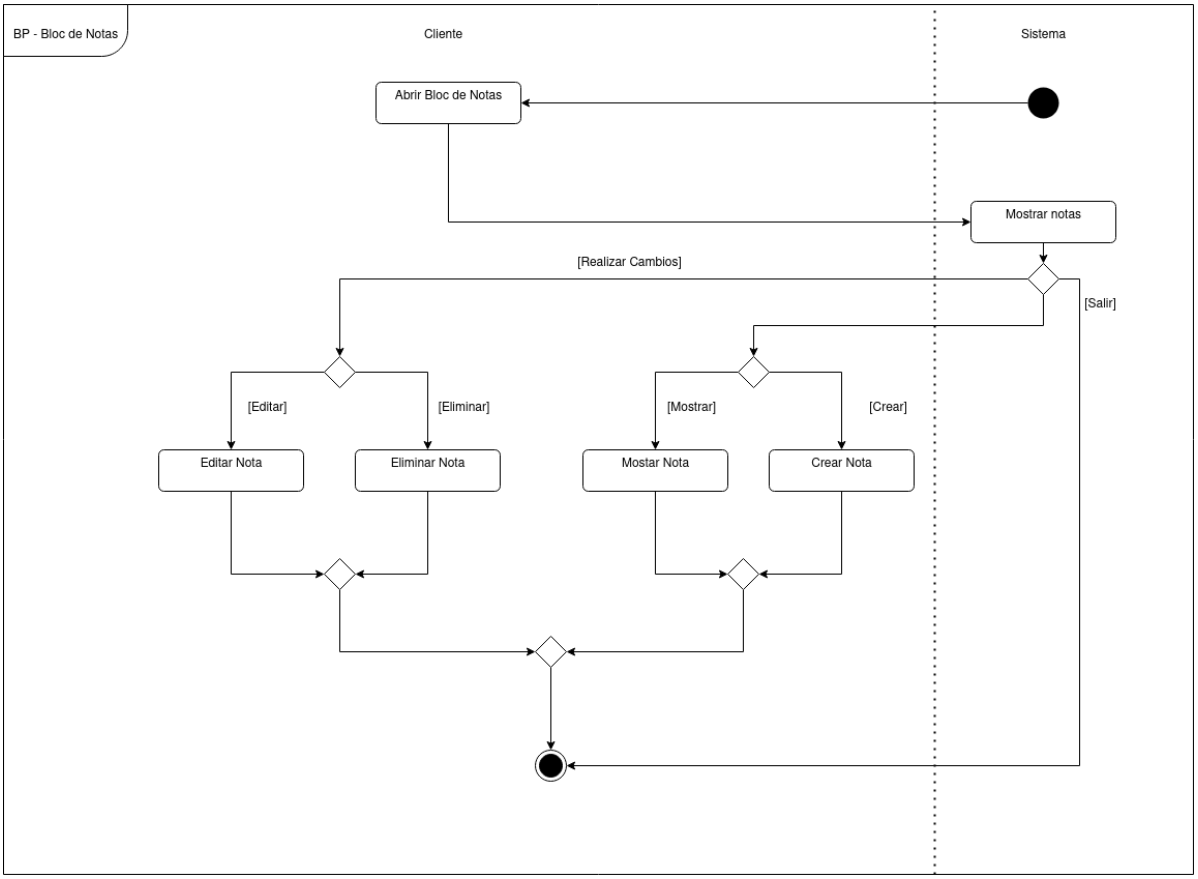


Figura 8.5: Diagrama de Actividad Gestión del Bloc de Notas

8.2.3. Especificación de Casos de Uso

Nombre	**Iniciar sesión**
ID	1
Descripción	El objetivo del caso de uso es permitir a un usuario iniciar sesión en la aplicación
Actor principal	Usuario
Actor secundario	Ninguno
Precondición	La cuenta debe de existir
Flujo principal	1) El caso de uso comienza cuando el usuario selecciona “Iniciar sesión”. 2) El sistema pide al usuario su login y contraseña. 3) El usuario introduce los datos. 4) El sistema valida los datos. 5) El sistema inicia la sesión del usuario.
Postcondición	La sesión queda iniciada.
Flujo alternativo	Cancelar. Error en el login o contraseña.

Cuadro 8.2: Especificación Caso de Uso: Asignar espacio comercial

Nombre	**Cerrar sesión**
ID	2
Descripción	El objetivo del caso de uso es permitir a un usuario cerrar sesión en la aplicación
Actor principal	Usuario
Actor secundario	Ninguno
Precondición	La sesión debe de estar iniciada
Flujo principal	1) El caso de uso comienza cuando el usuario selecciona “Cerrar sesión”. 2) El sistema pide confirmación para cerrar la sesión. 3) El usuario confirma el cierre de sesión. 4) El sistema cierra la sesión del usuario.
Postcondición	La sesión queda cerrada.
Flujo alternativo	Cancelar.

Cuadro 8.3: Especificación Caso de Uso: Cerrar sesión

Nombre	**Registrarse**
ID	3
Descripción	El objetivo del caso de uso es permitir a un usuario sin registrar registrarse en la aplicación.
Actor principal	Usuario sin registrar
Actor secundario	Ninguno
Precondición	La cuenta no debe de existir
Flujo principal	1) El caso de uso comienza cuando el usuario sin registrar selecciona “Registrarse”. 2) El sistema pide los datos al usuario sin registrar. 3) El usuario sin registrar introduce sus datos para crear la cuenta. 4) El sistema valida los datos introducidos. 5) El sistema crea la nueva cuenta.
Postcondición	La cuenta queda creada.
Flujo alternativo	Cancelar. Error en los datos. Cuenta ya existente.

Cuadro 8.4: Especificación Caso de Uso: Registrarse

Nombre	**Ver contactos**
ID	4
Descripción	El objetivo del caso de uso es permitir a un usuario ver todos los contactos que tenga guardados.
Actor principal	Usuario
Actor secundario	Ninguno
Precondición	Ninguna
Flujo principal	1) El caso de uso comienza cuando el usuario selecciona ”Ver contactos”. 2) El sistema recupera todos los contactos guardados del usuario. 3) El sistema muestra un listado de todos los contactos. Extension Point: Ver información del contacto.
Postcondición	Se muestran todos los contactos guardados.
Flujo alternativo	Ninguno.

Cuadro 8.5: Especificación Caso de Uso: Ver contactos

Nombre	**Ver información del contacto**
ID	5
Descripción	El objetivo del caso de uso es permitir a un usuario ver la información guardada de un contacto.
Actor principal	Usuario
Actor secundario	Ninguno
Precondición	El contacto debe de existir
Flujo principal	1) El caso de uso comienza cuando el usuario selecciona “Ver información del contacto”. 2) El sistema muestra la información del contacto seleccionado, describiendo todos los campos del contacto.
Postcondición	Se muestra la información del contacto.
Flujo alternativo	Ninguno.

Cuadro 8.6: Especificación Caso de Uso: Ver información del contacto

Nombre	**Buscar contacto**
ID	6
Descripción	El objetivo del caso de uso es permitir a un usuario busca un contacto de entre los que tiene guardados.
Actor principal	Usuario
Actor secundario	Ninguno
Precondición	Ninguna
Flujo principal	1) El caso de uso comienza cuando el usuario selecciona “Buscar contacto”. 2) El sistema pide al usuario el nombre del contacto a buscar. 3) El usuario introduce el nombre del contacto a buscar. 4) El sistema muestra el contacto buscado.
Postcondición	Muestra el resultado de la búsqueda.
Flujo alternativo	No se encontró el contacto.

Cuadro 8.7: Especificación Caso de Uso: Buscar contacto

Nombre	**Añadir contacto**
ID	7
Descripción	El objetivo del caso de uso es permitir a un usuario añadir un nuevo contacto.
Actor principal	Usuario
Actor secundario	Ninguno
Precondición	No debe de existir el contacto
Flujo principal	1) El caso de uso comienza cuando el usuario selecciona “Añadir contacto”. 2) El sistema pide los datos al usuario. 3) El usuario introduce los datos del nuevo contacto que quiere añadir. 4) El sistema valida los datos. 5) El sistema pide confirmación para guardar el contacto. 6) El usuario confirma la creación del contacto. 7) El sistema guarda el contacto.
Postcondición	El contacto queda añadido a la lista de contactos.
Flujo alternativo	Error en los datos del nuevo contacto. Cancelar.

Cuadro 8.8: Especificación Caso de Uso: Añadir contacto

Nombre	**Eliminar contacto**
ID	8
Descripción	El objetivo del caso de uso es permitir a un usuario eliminar un contacto.
Actor principal	Usuario
Actor secundario	Ninguno
Precondición	Debe de existir el contacto
Flujo principal	1) El caso de uso comienza cuando el usuario selecciona “Eliminar contacto”. 2) El sistema pide al usuario el nombre del contacto a buscar. 3) El usuario introduce el nombre del contacto a eliminar. 4) Include (Buscar contacto). 5) El sistema pide confirmación de borrado del contacto. 6) El usuario confirma el borrado del contacto. 7) El sistema elimina al contacto.
Postcondición	El contacto queda eliminado de la lista de contactos.
Flujo alternativo	No se encontró el contacto. Cancelar.

Cuadro 8.9: Especificación Caso de Uso: Eliminar contacto

Nombre	**Actualizar contacto**
ID	9
Descripción	El objetivo del caso de uso es permitir a un usuario actualizar los datos guardados de un contacto determinado.
Actor principal	Usuario
Actor secundario	Ninguno
Precondición	Debe de existir el contacto a modificar.
Flujo principal	1) El caso de uso comienza cuando el usuario selecciona “Actualizar contacto”. 2) Include (Buscar contacto). 3) El usuario modifica los campos deseados. 4) El sistema pedirá confirmación para actualizar los datos del contacto. 5) El usuario confirma la actualización de los datos. 5) El usuario confirma la actualización de los datos. 6) El sistema modifica en la base de contactos los campos del contacto en cuestión.
Postcondición	La información del contacto queda actualizada.
Flujo alternativo	Cancelar.

Cuadro 8.10: Especificación Caso de Uso: Actualizar contacto

Nombre	**Importar contacto**
ID	10
Descripción	El objetivo del caso de uso es permitir a un usuario importar contactos desde otras aplicaciones.
Actor principal	Usuario
Actor secundario	Ninguno
Precondición	Ninguna.
Flujo principal	1) El caso de uso comienza cuando el usuario selecciona “Importar contacto”. 2) El sistema mostrará las aplicaciones disponibles a través de las cuales importar un contacto. 3) El usuario selecciona la aplicación en cuestión. 4) El sistema importa el contacto en cuestión a través de dicha aplicación.
Postcondición	El contacto ha sido añadido a la agenda .
Flujo alternativo	**Error al importar contacto** **Contacto ya existente:** 4.A1) El sistema notifica al usuario que ya existe el contacto. 4.A2) Si el usuario selecciona sobreescribir contacto. 4.A2.1) Include Borrar Contacto. 4.A3) Include Añadir Contacto.

Cuadro 8.11: Especificación Caso de Uso: Importar contacto

Nombre	**Exportar contacto**
ID	11
Descripción	El objetivo del caso de uso es permitir comprimir un contacto en un fichero de extensión compatible para traspasarlo a un dispositivo diferente.
Actor principal	Usuario
Actor secundario	Ninguno
Precondición	El usuario ha seleccionado a un contacto existente para exportar.
Flujo principal	1) El caso de uso comienza cuando el usuario selecciona “Exportar contacto”. 2) El sistema creará un fichero con la información del contacto. 3) El sistema mostrará las aplicaciones posibles a través de las cuales enviarlo. 4) El usuario seleccionará una aplicación para enviar al contacto. 5) El sistema enviará el fichero del contacto al destinatario a través de la aplicación seleccionada.
Postcondición	El contacto ha sido exportado de forma correcta.
Flujo alternativo	Error en la exportación.

Cuadro 8.12: Especificación Caso de Uso: Exportar contacto

Nombre	**Exportar contacto Avanzado**
ID	12
Descripción	El objetivo del caso de uso es permitir comprimir todos los contactos en un fichero de extensión compatible para traspasarlo a un dispositivo diferente.
Actor principal	Usuario
Actor secundario	Ninguno
Precondición	Existe al menos un contacto.
Flujo principal	1) El caso de uso comienza cuando el usuario selecciona “Exportar contacto Avanzado”. 2) El sistema mostrará la lista de contactos a exportar. 3) Si el usuario selecciona un número menor del número total de contactos. 3.1) El sistema creará un fichero con la información del grupo de contacto 4) Sino, Si el usuario clica ***Seleccionar todos los contactos***. 4.1) El sistema creará un fichero con la información de todos los contactos de la agenda. 5) El sistema mostrará las aplicaciones posibles a través de las cuales enviarlo. 6) El usuario seleccionará una aplicación para enviar al contacto. 7) El sistema enviará el fichero del contacto al destinatario a través de la aplicación seleccionada.
Postcondición	El grupo de contactos ha sido exportado de forma correcta.
Flujo alternativo	Error en la exportación.

Cuadro 8.13: Especificación Caso de Uso: Exportar contacto Avanzado

Nombre	**Visualizar calendario de eventos**
ID	13
Descripción	El objetivo del caso de uso es permitir a un usuario visualizar su calendario de eventos.
Actor principal	Usuario
Actor secundario	Ninguno
Precondición	Ninguna.
Flujo principal	1) El caso de uso comienza cuando el usuario selecciona “Visualizar calendario de eventos”. 2) El sistema recopila todos los eventos guardados del usuario. 3) El sistema muestra todos los eventos en su fecha correspondiente en el calendario.
Postcondición	Se muestra el calendario de eventos.
Flujo alternativo	Ninguno.

Cuadro 8.14: Especificación Caso de Uso: Visualizar calendario de eventos

Nombre	**Añadir evento**
ID	14
Descripción	El objetivo del caso de uso es permitir a un usuario añadir un evento a su calendario.
Actor principal	Usuario
Actor secundario	Ninguno
Precondición	Ninguna.
Flujo principal	1) El caso de uso comienza cuando el usuario selecciona “Añadir evento”. 2) El sistema pide fecha para el evento. 3) El usuario introduce la fecha para el evento. 4) El sistema pide información sobre el nuevo evento. 5) El usuario introduce la información del evento. 6) El sistema pide confirmación para crear el evento. 7) El usuario confirmación para crear el evento. 8) El sistema guarda el evento en el calendario.
Postcondición	El evento queda añadido al calendario.
Flujo alternativo	Cancelar. Error en la fecha.

Cuadro 8.15: Especificación Caso de Uso: Añadir evento

Nombre	**Eliminar evento**
ID	15
Descripción	El objetivo del caso de uso es permitir a un usuario eliminar un evento de su calendario.
Actor principal	Usuario
Actor secundario	Ninguno
Precondición	El evento debe de existir.
Flujo principal	1) El caso de uso comienza cuando el usuario selecciona “Eliminar evento”. 2) El sistema pide confirmación para borrar el evento. 3) El usuario confirma el borrado del evento. 4) El sistema borra el evento del calendario.
Postcondición	El evento queda eliminado del calendario.
Flujo alternativo	Cancelar.

Cuadro 8.16: Especificación Caso de Uso: Eliminar evento

Nombre	**Visualizar lista de tareas**
ID	17
Descripción	El objetivo del caso de uso es permitir a un usuario ver su lista de tareas por hacer.
Actor principal	Usuario
Actor secundario	Ninguno
Precondición	Ninguna.
Flujo principal	1) El caso de uso comienza cuando el usuario selecciona “Ver lista de tareas”. 2) El sistema recupera todas las tareas del usuario. 3) El sistema muestra todas las tareas de la lista al usuario.
Postcondición	Se muestran todas las tareas de la lista.
Flujo alternativo	Ninguno.

Cuadro 8.17: Especificación Caso de Uso: Visualizar lista de tareas

Nombre	**Crear lista de tareas**
ID	18
Descripción	El objetivo del caso de uso es permitir a un usuario crear una nueva lista de tareas.
Actor principal	Usuario
Actor secundario	Ninguno
Precondición	Ninguna.
Flujo principal	1) El caso de uso comienza cuando el usuario selecciona “Crear lista de tareas”. 2) El sistema pide al usuario las tareas para añadir a la nueva lista. 3) El usuario introduce las tareas en la lista. 4) El sistema guarda la nueva lista con las tareas del usuario.
Postcondición	La lista de tareas queda creada.
Flujo alternativo	Cancelar.

Cuadro 8.18: Especificación Caso de Uso: Crear lista de tareas

Nombre	**Borrar lista de tareas**
ID	19
Descripción	El objetivo del caso de uso es permitir a un usuario borrar una lista de tareas ya creada.
Actor principal	Usuario
Actor secundario	Ninguno
Precondición	La lista debe de existir.
Flujo principal	1) El caso de uso comienza cuando el usuario selecciona “Borrar lista de tareas”. 2) El usuario selecciona la lista de tareas a borrar. 3) El sistema pide confirmación de borrado de la lista. 4) El usuario confirma el borrado de la lista de tareas. 5) El sistema borra la lista de tareas.
Postcondición	La lista de tareas queda eliminada.
Flujo alternativo	Cancelar.

Cuadro 8.19: Especificación Caso de Uso: Borrar lista de tareas

Nombre	**Modificar lista de tareas**
ID	20
Descripción	El objetivo del caso de uso es permitir a un usuario modificar las tareas de una lista de tareas.
Actor principal	Usuario
Actor secundario	Ninguno
Precondición	La lista de tareas debe de estar creada.
Flujo principal	1) El caso de uso comienza cuando el usuario selecciona “Modificar lista de tareas”. 2) El usuario modifica los campos de la lista de tareas según le parezca conveniente. 3) El sistema pide confirmación de las modificaciones. 4) El usuario confirma los cambios. 5) El sistema guarda la lista de tareas modificada.
Postcondición	La información de las tareas de la lista quedan actualizada.
Flujo alternativo	Cancelar.

Cuadro 8.20: Especificación Caso de Uso: Modificar lista de tareas

Nombre	**Ver bloc de notas**
ID	21
Descripción	El objetivo del caso de uso es permitir a un usuario ver su bloc de notas.
Actor principal	Usuario
Actor secundario	Ninguno
Precondición	Debe existir el bloc de notas.
Flujo principal	1) El caso de uso comienza cuando el usuario selecciona “Ver bloc de notas”. 2) El sistema muestra al usuario el bloc de notas
Postcondición	Se muestra el bloc de notas del usuario.
Flujo alternativo	Ninguno.

Cuadro 8.21: Especificación Caso de Uso: Ver bloc de notas

Nombre	**Modificar bloc de notas**
ID	22
Descripción	El objetivo del caso de uso es permitir a un usuario modificar su bloc de notas.
Actor principal	Usuario
Actor secundario	Ninguno
Precondición	Debe existir el bloc de notas.
Flujo principal	1) El caso de uso comienza cuando el usuario selecciona “Modificar bloc de notas”. 2) El sistema muestra al usuario el bloc de notas. 3) El usuario introduce las modificaciones pertinentes. 4) El sistema pide confirmación para guardar los cambios. 5) El usuario confirma los cambios. 6) El sistema guarda los cambios.
Postcondición	Se actualiza el bloc de notas del usuario.
Flujo alternativo	Cancelar.

Cuadro 8.22: Especificación Caso de Uso: Modificar bloc de notas

Nombre	**Llamar a contacto**
ID	23
Descripción	El objetivo del caso de uso es permitir a un usuario llamar a uno de sus contactos de la agenda.
Actor principal	Usuario
Actor secundario	Ninguno
Precondición	Debe existir el contacto al que llamar.
Flujo principal	1) El caso de uso comienza cuando el usuario selecciona “Llamar a contacto”. 2) El sistema abrirá el menú de opciones de la llamada y llamará al contacto.
Postcondición	Se llamará al contacto seleccionado.
Flujo alternativo	Cancelar llamada.

Cuadro 8.23: Especificación Caso de Uso: Llamar a contacto

9 Prototipado

9.1. ¿Qué es un Prototipo?

El prototipado es una versión inicial de la idea de un producto o servicio. El prototipado nos permite probar, evaluar y validar si la idea que tenemos en mente cumple los objetivos impuestos por el usuario. Gracias a este prototipo podremos validar esas ideas que tenemos de productos o servicios pero que no sabemos cómo reflejarlas ni cómo llevarlas a cabo.

El prototipo nos ayuda a tener una herramienta física con la que poder validar que esa idea tiene sentido.

Por otro lado, también nos sirve para estimar costes. Es decir, tú tienes un prototipo y acudirás a un equipo de desarrollo con él y el equipo de desarrollo te dirá cuál será el alcance del producto.

9.2. Tipos de Prototipos

9.2.1. Sketching

Es el primer dibujo hecho a lápiz o bolígrafo en papel. Se trata de un primer boceto que ayudará a generar nuevas ideas y ver con claridad los pain point de los usuarios. En esta fase, el diseño aún está alejado del producto final y debido a su bajo coste de producción es el prototipo que más cambios puede recibir.

9.2.2. Prototipo de baja fidelidad (Wireframes)

Si con los sketchings perseguimos ante todo la generación de ideas, con los wireframes nos centraremos en el diseño del contenido. Normalmente se hacen en escala de grises o en blanco y negro. Con este tipo de prototipo veremos los bocetos anteriores enriquecidos.

9.2.3. Prototipo de media fidelidad (Mockups)

Se trata de la parte visual del proyecto y debe contener amplitud de detalles como los colores, tipografía, contenido, etc. Es el prototipo que mejor representa el producto final aunque faltaría incluir las interacciones que se realizarán en la fase siguiente.

9.2.4. Prototipo de alta fidelidad (Maqueta)

Con la maqueta validamos si la idea funciona o no. En este momento es cuando normalmente se perciben los problemas de usabilidad. Mediante los testeos de interactividad y los feedback de los usuarios, veremos todos esos detalles que necesitan una revisión. Es decir, se trata de la fase más determinante para validar la usabilidad de todo el producto.

9.3. Prototipos Desarrollados

Previo al desarrollo del proyecto decidimos crear varios prototipos para verificar las funcionalidades desarrolladas y planteadas por la empresa de tal forma que cumplan con las expectativas. La intención de estos prototipos es esclarecer la idea general del proyecto para facilitar el desarrollo del mismo.

A continuación, mostraremos los prototipos realizados. Estos no cubren todas las funcionalidades, pero nos ayudarán a tener una idea mucho más específica de como queremos que se vea nuestra aplicación y como funcionarán los diferentes servicios.

Además, gracias a estos prototipos, podremos darnos cuenta de cualquier servicio o detalle de la funcionalidad en especifico que sea necesario y no hayamos tenido en cuenta.

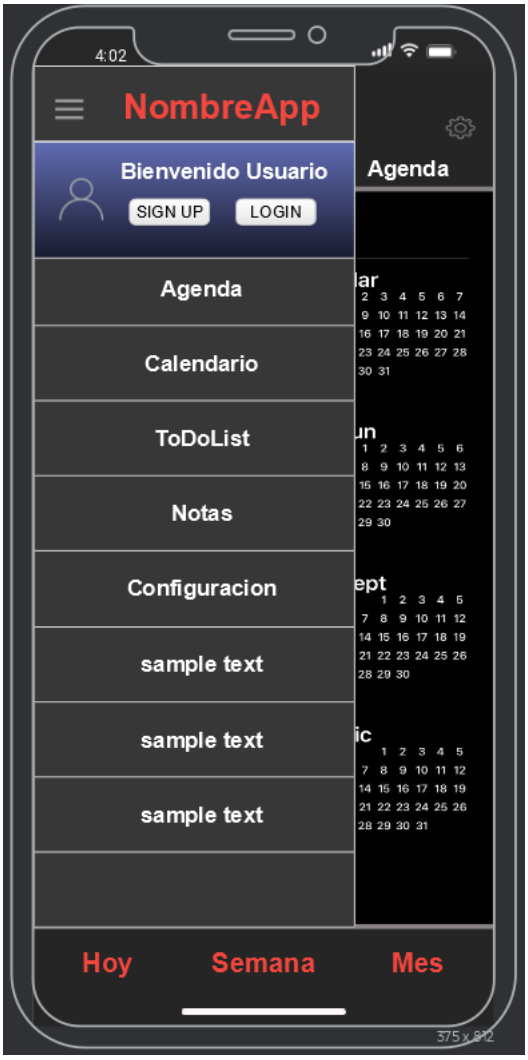
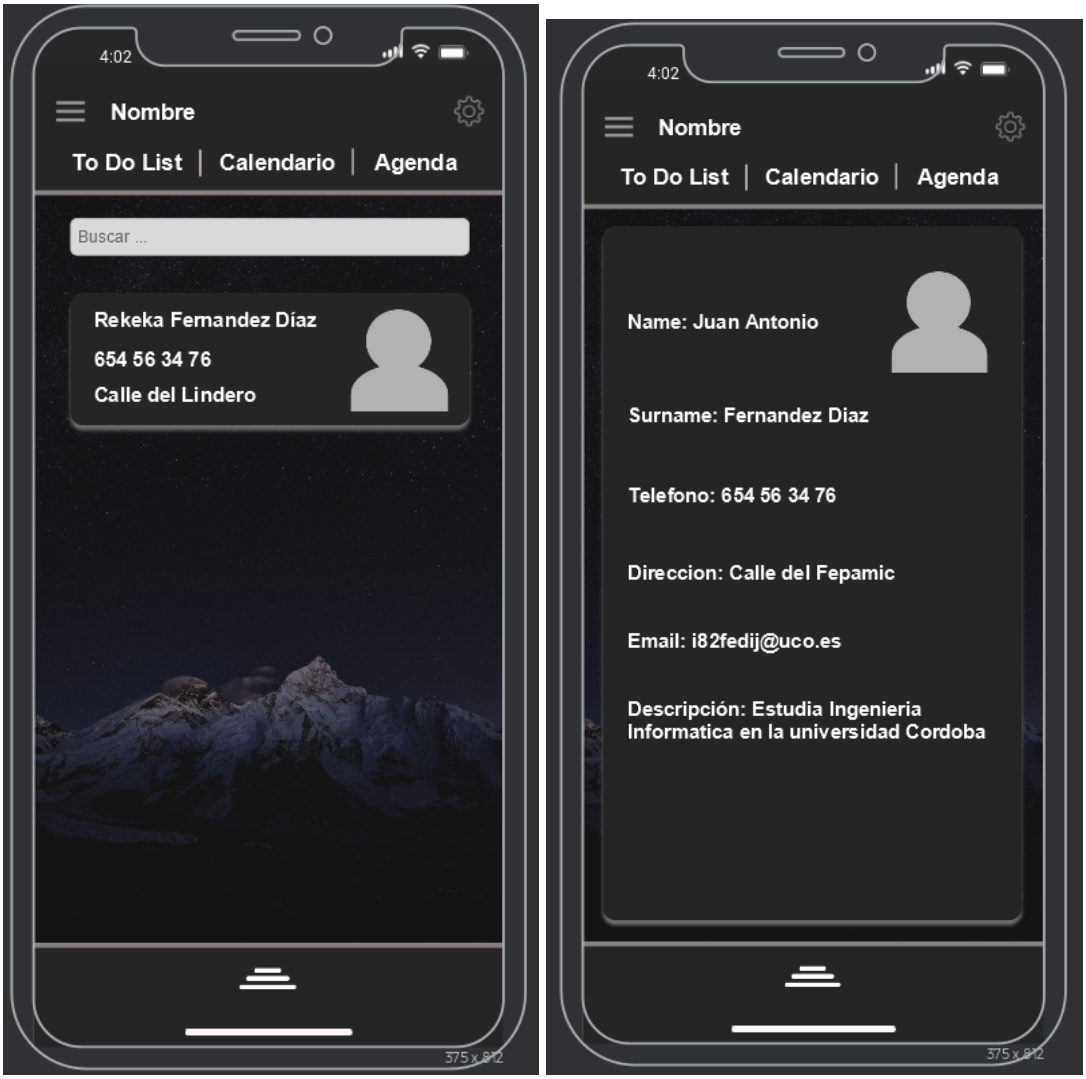


Figura 9.1: Prototipo de la Ventana Principal

(a) Agenda **(b)** Vista Contacto

Figura 9.2: Prototipos del Calendario



(a) Agenda **(b)** Vista Contacto

Figura 9.3: Prototipos del Calendario

10
Glosario

Android: Sistema Operativo diseñado para dispositivos de móviles. Android es un proyecto de código abierto y esta soportado y desarrollado por Open Handset Alliance (OHA).

Bluetooth: Es una tecnología de comunicación inalámbrica que facilita la transferencia de datos entre dispositivos (por ejemplo, teléfono móvil y un auricular inalámbrico).

Cifrado: Proceso de cifrar sistemáticamente la transmisión de una secuencia de bits para que una persona no autorizada no la pueda descifrar.

GPS: sistema de posicionamiento global es un sistema de navegación por radio basado en satélite que consiste en tres segmentos - la constelación de satélites, la red de control desde tierra y el equipo del usuario.

IP: Protocolo de Internet es un protocolo de comunicación que utiliza una técnica de intercambio-de-paquetes para transmitir datos sobre la Internet.

Sincronización: Consiste en la actualización de un conjunto de datos en dos ubicaciones. Los cambios en la información son copiados de una ubicación a otra y viceversa, de modo que ambas ubicaciones.

Wi-Fi: Proviene del término “Wireless Fidelity’’, consiste en una tecnología de transmisión de datos inalámbricos utilizada para Internet y que se basa en el estándar 802.11. Tecnología que permite la conexión inalámbrica entre dispositivos electrónicos, ordenadores, smartphones, tablets, televisores, videoconsolas, etc.

Accesibilidad: Posibilidad de acceso a los contenidos por cualquier usuario independiente de sus capacidades físicas o mentales.

App: Nombre usado para referirse a las aplicaciones móviles, que surge de la palabra application. Consiste en una pieza de software que se ejecuta en teléfonos móviles y tabletas.

Benchmarking: Consiste en el proceso sistemático para evaluar comparativamente productos, servicios y procesos. Consiste en el estudio de la usabilidad y los servicios ofrecidos.

Densidad de pantalla: Cantidad de píxeles por espacio físico que tiene una pantalla. Normalmente se mide en píxeles por pulgada. Las densidades son diferentes dependiendo del modelo de móvil.

Experiencia de Usuario: Concentra las emociones y percepciones que tiene una persona al usar una interfaz o producto. Es afectada por la accesibilidad, diseño visual, diseño de interacción, usabilidad y curva de aprendizaje.

Interfaz o UI: Consiste en la capa que existe entre el usuario y el dispositivo, que le permite interactuar con este último.

11

Identificación de componentes

Una vez realizada la especificación de requisitos y determinados los procesos de negocio y los casos de uso, debemos de identificar los diferentes componentes que formarán nuestra aplicación a desarrollar.

Para realizar esto se tendrá en cuenta una distinción entre la capa de Negocio y la capa del Sistema, para así más fácil el trabajo de desarrollo y controlar mejor las dependencias entre componentes.

Un aspecto que tenemos que tener en cuenta es que el BCM ya tratado anteriormente en la **Figura 8.1**, en esta etapa deberá de ser refinado indicando los campos de interés para cada uno de los "conceptos" importantes del negocio. Este refinado se puede apreciar en la **Figura 11.1**

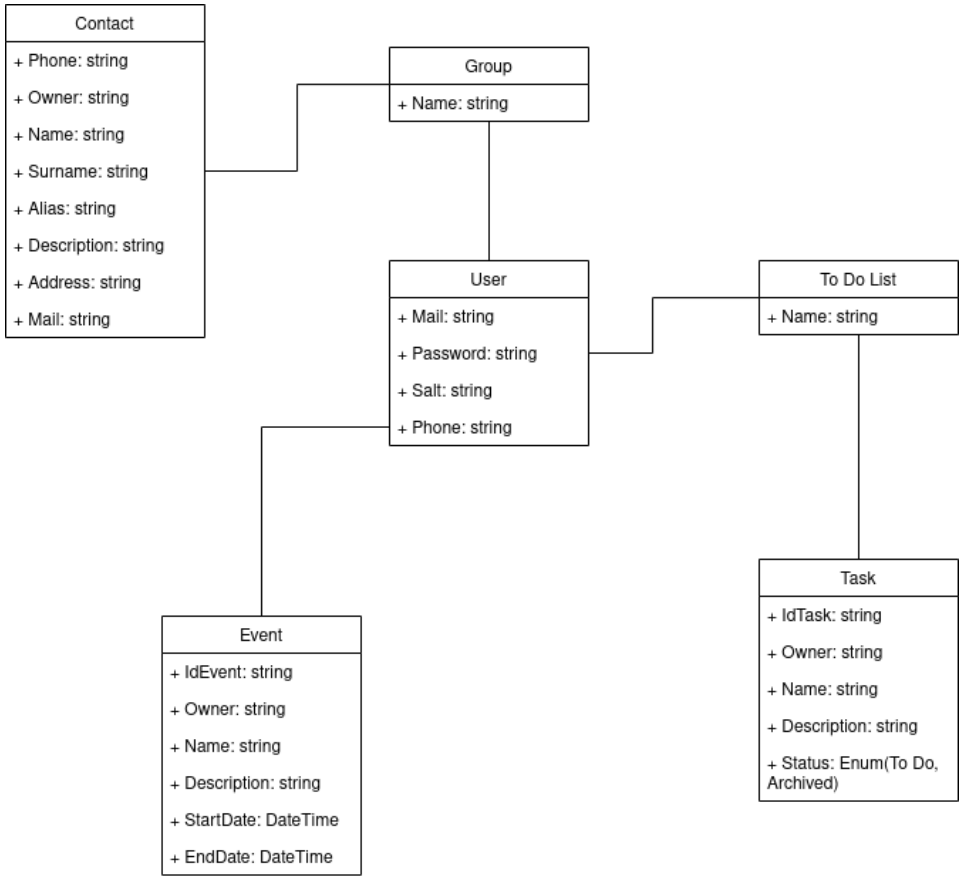


Figura 11.1: Business Type Model (Refinado del BCM)

11.1. Identificación de las Interfaces del Sistema

11.1.1. Funciones para el caso de uso: Iniciar sesión, cerrar sesión, registrarse

- Solicitar usuario y contraseña
- Validar usuario y contraseña
- Enviar correo de confirmación

11.1.2. Funciones para el caso de uso: Ver contactos, Ver información del contacto, Buscar contacto

- Obtener listado de contactos
- Obtener información de contacto
- Buscar contacto
- Introducir datos de búsqueda
- Filtrar contactos

11.1.3. Funciones para el caso de uso: Añadir contacto, eliminar contacto, modificar contacto

- Introducir datos
- Validar datos
- Buscar contacto
- Registrar contacto
- Borrar contacto
- Mostrar información del contacto
- Actualizar contacto

11.1.4. Funciones para el caso de uso: Visualizar calendario de eventos

- Mostrar calendario
- Seleccionar Vista Diaria/Mensual/Anual

11.1.5. Funciones para el caso de uso: Añadir evento, eliminar evento, actualizar evento

- Introducir datos
- Validar datos
- Registrar evento
- Borrar evento
- Actualizar evento
- Mostrar información del evento
- Buscar evento

11.1.6. Funciones para el caso de uso: Visualizar lista de tareas

- Mostrar listas de tareas
- Mostrar tareas de la lista

11.1.7. Funciones para el caso de uso: Crear lista de tareas, borrar lista de tareas, modificar lista de tareas

- Introducir datos
- Registrar tarea
- Borrar tarea
- Modificar tarea
- Mostrar tarea

- Registrar lista de tareas
- Borrar lista de tareas
- Modificar lista de tareas
- Mostrar lista de tareas

11.1.8. Funciones para el caso de uso: Ver bloc de notas, Modificar bloc de notas

- Mostrar bloc de notas
- Modificar bloc de notas

Las interfaces obtenidas serían:

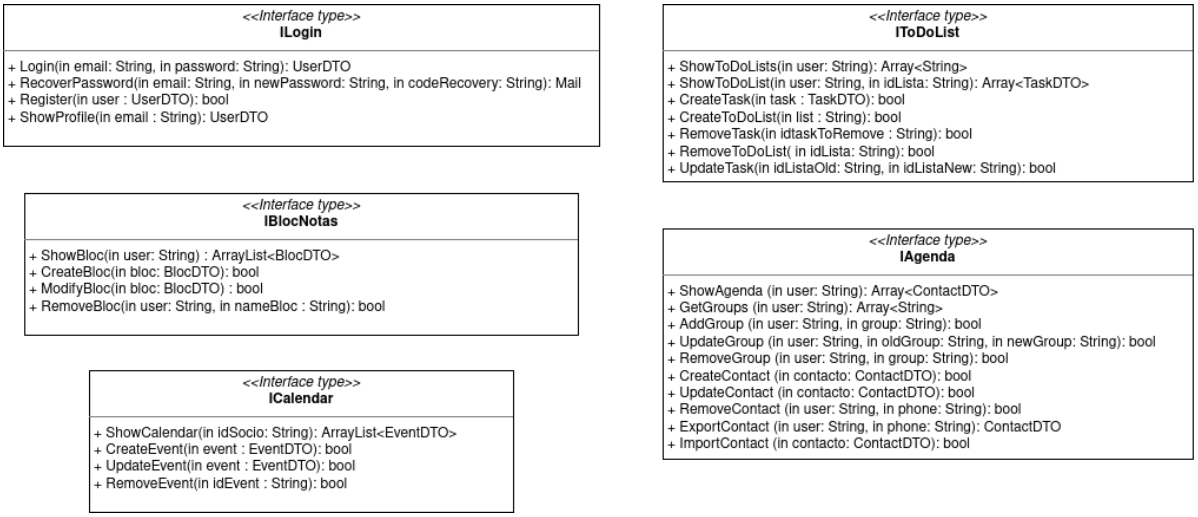


Figura 11.2: Interfaces del Sistema

11.2. Identificación de las Interfaces de Negocio

A la hora de trabajar con la capa de negocio, deberemos de tener en cuenta cuales son los posibles datos con los cuales trabajaremos para ello utilizaremos una base de datos con la cual interactuaremos a través de las siguientes interfaces.

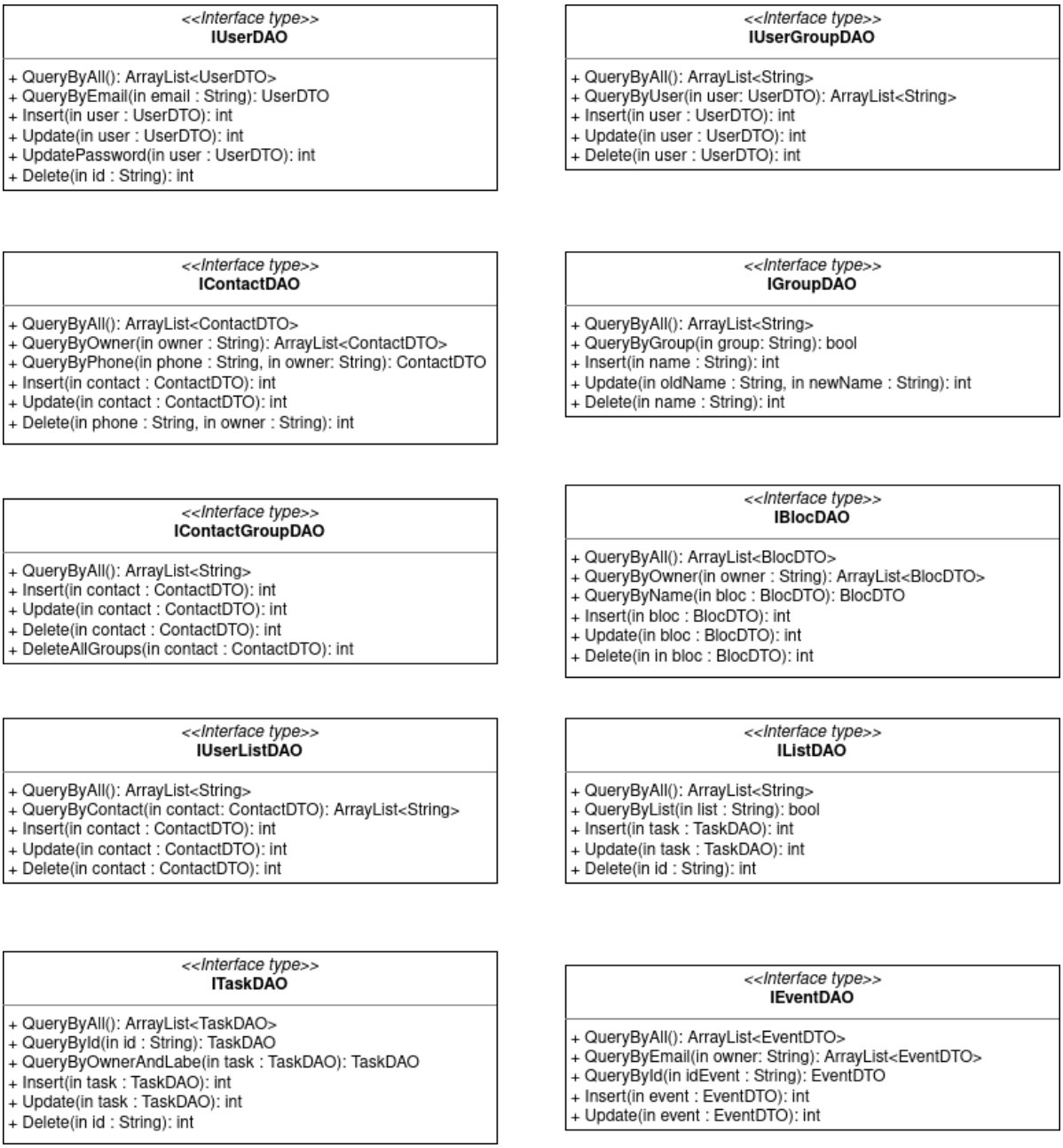


Figura 11.3: Interfaces del Negocio

Tras ver las interfaces debemos de comprender que la base de datos con la cual trabajamos sigue el siguiente esquema relacional:

Una vez comprendido todo esto podemos pasar a detecar los diferentes componentes de los cuales se componen nuestra aplicación.

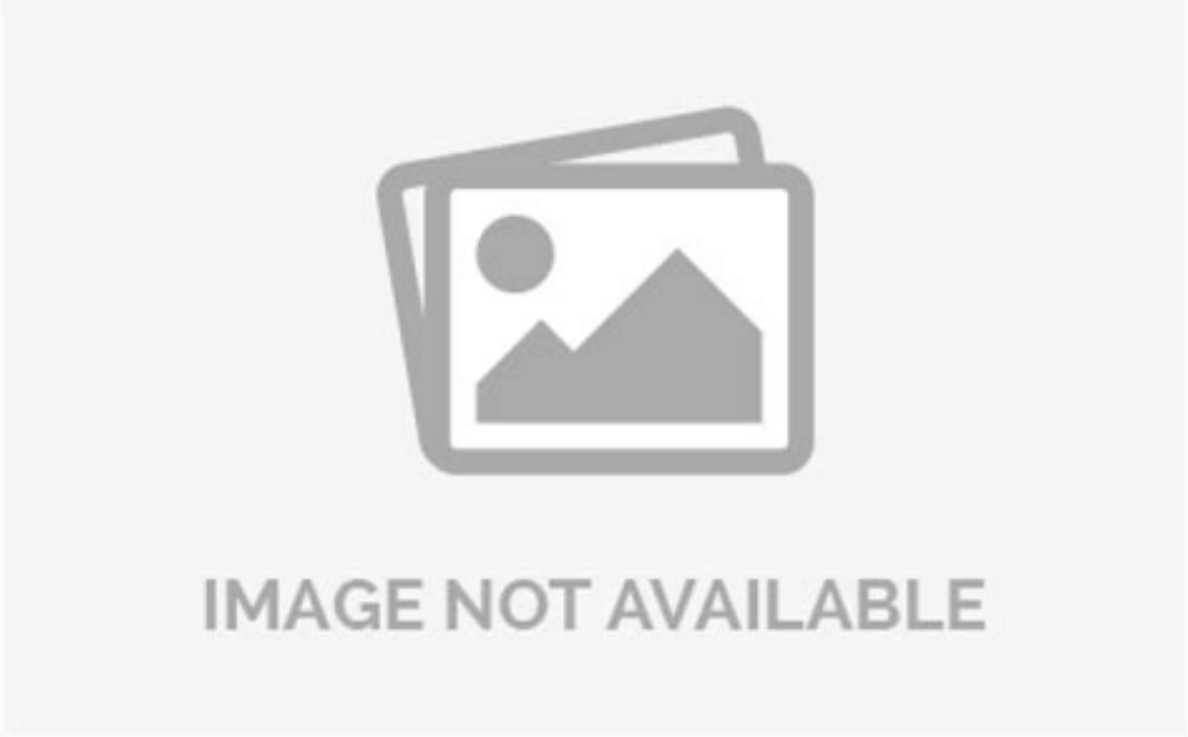


Figura 11.4: Diagrama Entidad Relación Base de Datos

11.3. Identificación de Componentes del Sistema

En cuanto a los componentes del sistema podemos ver rápidamente como tenemos 5 componentes los cuales se encargarán de implementar su interfaz, debido a que estamos desarrollando la aplicación como una aplicación Modelo-Vista-Controlador, debemos de comprender que en cada componente tendremos un controlador el cual implementará cada función de las interfaces anteriormente nombradas, de esta forma el componente constaría de un conjunto de **Servlet**, los cuales proporcionan estas funcionalidades.

Una vez comprendido esto podemos especificar cuales serían los componentes del sistema encargados de ofrecer nuestros servicios:

- **Login:** Este componente implementara la interfaz del Sistema de **ILogin**, siendo necesarios 4 controladores, para cumplirla.
- **Agenda:** Este componente implementara la interfaz del Sistema de **IAgenda**.
- **Bloc:** Este componente implementara la interfaz del Sistema de **IBlocNotas**.
- **Calendar:** Este componente implementara la interfaz del Sistema de **ICalendar**.
- **ToDoList:** Este componente implementara la interfaz del Sistema de **IToDoList**.

Estos serían los componentes del sistema que ofrecerán los servicios a nuestros clientes de forma que facilmente desde las vistas podrán acceder a los “datos” que se encuentran en nuestros servidores.

11.4. Identificación de Componentes de Negocio

En cuanto a los componentes de Negocio, debemos de comprender que estos serán todos aquellos relacionados con la generación de los datos y su tratamiento, por ende, se generará un componente por cada interfaz de Negocio.

Un aspecto que nos gustaría destacar sería el uso de un componente ya desarrollado con anterioridad en la compañía que será reutilizado el cual será encargado del envió de correos a los usuarios a la hora de registrarse y en los momentos de **recuperación de contraseña**, para ello utilizaremos el componente **Mail** el cual será utilizado en las funcionalidades del Sistema de **Register y RecoverPassword**.

11.5. Arquitectura Final del Sistema

En cuanto a la arquitectura final del sistema a desarrollar podemos ver como poseemos una alta cantidad de componentes de Negocio los cuales serán los Módelos de la aplicación encargados de interaccionar con los Controladores (Componentes del Sistema), para así ofrecer la información que soliciten las Vistas (desarrolladas con Ionic y Angular). El esquema de la arquitectura del Backend se puede apreciar en la **Figura 11.5**

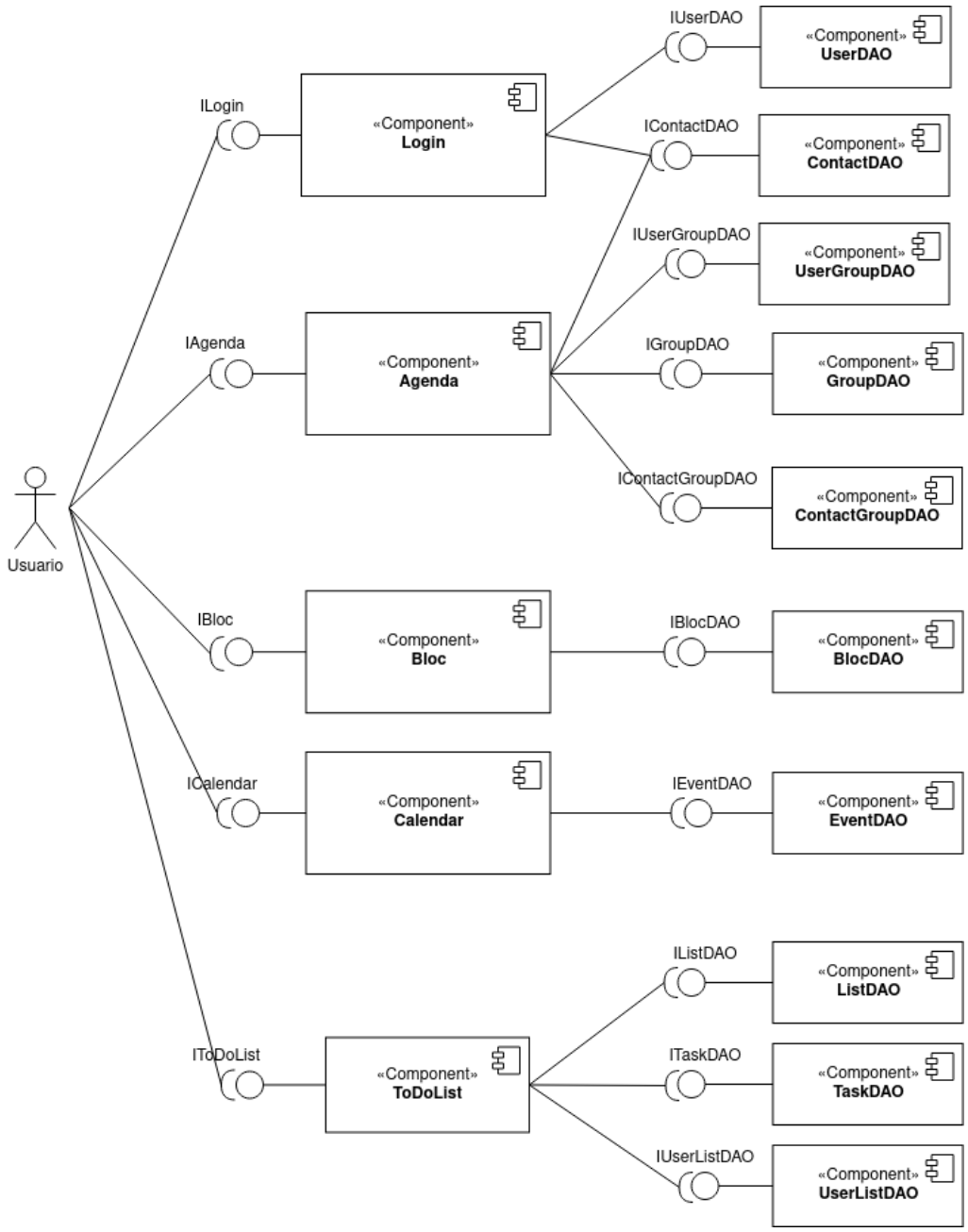


Figura 11.5: Arquitectura Final del Sistema

12 Implementación

Debido a que el proyecto que se encuentra en desarrollo se pretende que sea una aplicación móvil debemos de tener en cuenta las tecnologías que vamos a utilizar para el desarrollo.

En primera lugar expondremos las herramientas con las cuales se trabajará de forma que quede todo expuesto y concreto.

12.1. Tecnologías Utilizadas

Para el backend como se ha comentado anteriormente se desarrollarán en Java los diferentes controladores para cada uno de los componentes del sistema, para ello utilizaremos la clase **Servlet**, en los cuales haremos uso de los DAO los cuales serán desarrollados como intermediario con la base de datos. De forma que mediantes los DTO(Data Transfer Object) encapsularemos las comunicaciones con la base de datos.

Esta base de datos sera implementada mediante **MySQL** la cual sigue el modelo relacional propuesto en la **sección 11.4** .

Esta base de datos se encontrara instalada y hosteada por la Raspberry Pi 4 que ya indicamos en el capítulo de **Recursos (4)**.

Para ejecutar y permitir ofrecer el servicio de los controladores utilizaremos el servidor web de **Apache Tomcat** el cual en concreto utilizaremos la versión 9. La cual ha sido ya utilizada anteriormente en otros proyectos.

Como sabemos para utilizar el servidor web de Tomcat debemos de especificar la versión de Java con la cual se desarrollará el proyecto. Actualmente la compañía trabaja con la versión estable de **Java 16 development** para la fase de desarrollo y para el despliegue la 16 stable.

Con todo esto expuesto podemos pasar ya a los aspectos del Frontend, en este caso debido a que nuestro objetivo es desarrollar una aplicación móvil, el grupo de desarrolladores Frontend decidieron introducir una tecnología que anteriormente no habíamos utilizado. Esta sería **Ionic** en concreto utilizaremos **Angular** como lengua de desarrollo para el Frontend.

Además de todas herramientas decidimos utilizar una instancia de **Heroku** para realizar un proxi, encargado de las peticiones de nuestros clientes, permitiendo así mejorar la seguridad de nuestros clientes.

12.2. Desarrollo de trabajo

A la hora de trabajar debido a que todos los miembros del grupo de desarrollo deben de tocar diferentes aspectos y trabajar en común se desarrollo un repositorio en **Github** en el cual se encontrarán todos los archivos necesarios para compilar la aplicación, probarla e instalarla.

13
Conclusiones

Como podemos ver en este proyecto se ha planteado desarrollar una aplicación móvil la cual nos permitiera agrupar distintos servicios que normalmente nos encontramos en distintas aplicaciones en nuestros móviles, de forma que desde una sola interfaz y con un formato amigable se pueda tener acceso a todos estos servicios.

Uno de los aspectos que podríamos destacar sería el gran aumento de conocimientos que han obtenido los miembros del proyecto, desde aspectos de la gestión de servidores, hosting, PaaS hasta aspectos en el desarrollo de aplicaciones móviles (Angular, Ionic, uso de otras tecnologías, etc).

Con todo ello pretendemos que con este documento y todo el código que se encuentra en el repositorio de Github de la Compañía de **Hokuro-Inc**, quede presente el trabajo desarrollado e implementado por parte de los miembros integrantes.

14

Bibliografía

1. Java Archive Downloads - Java SE 16. (2021). Retrieved 9 November 2021, from https://www.oracle.com/java/technologies/javase/jdk16-archive-downloads.html
2. Angular. (2021). Retrieved 9 November 2021, from https://angular.io/
3. JavaScript | MDN. (2021). Retrieved 9 November 2021, from https://developer.mozilla.org/es/docs/Web/JavaScript
4. HTML: Lenguaje de etiquetas de hipertexto | MDN. (2021). Retrieved 9 November 2021, from https://developer.mozilla.org/es/docs/Web/HTML
5. CSS | MDN. (2021). Retrieved 9 November 2021, from https://developer.mozilla.org/es/docs/Web/CSS
6. MySQL :: Download MySQL Community Server. (2021). Retrieved 9 November 2021, from https://dev.mysql.com/downloads/mysql/8.0.html
7. Apache Tomcat - Apache Tomcat 9 Software Downloads. (2021). Retrieved 9 November 2021, from https://tomcat.apache.org/download-90.cgi
8. UI Components | Ionic Documentation. (2021). Retrieved 13 November 2021, from https://cutt.ly/VYD9FKy
9. App closes when writing to an NFC tag · Issue #328 · chariotsolutions/phonegap-nfc. (2021). Retrieved 13 November 2021, from https://github.com/chariotsolutions/phonegap-nfc/issues/328
10. Unable to resolve dependency tree error when installing npm packages. (2021). Retrieved 13 November 2021, from https://cutt.ly/DYD9KKK
11. Dark Mode | Ionic Documentation. (2021). Retrieved 13 November 2021, from https://cutt.ly/vYD94XW
12. Ionic, W. (2021). How to hide the white screen after splash screen ionic 4 and capacitor. Retrieved 13 November 2021, from https://cutt.ly/NYD9VoC
13. Android Development | Ionic Documentation. (2021). Retrieved 16 November 2021, from https://ionicframework.com/docs/developing/android
14. Problemas de CORS y algunas soluciones para Ionic - DIGITAL55. (2021). Retrieved 16 November 2021, from https://cutt.ly/fYD91iz
15. GitHub - chariotsolutions/phonegap-nfc: PhoneGap NFC Plugin. (2021). Retrieved 16 November 2021, from https://github.com/chariotsolutions/phonegap-nfc#nfcreadermode
16. (2021). Retrieved 16 November 2021, from https://como-programar.net/ionic/nfc/
17. HttpServletRequest (Java EE 6). (2021). Retrieved 20 November 2021, from https://docs.oracle.com/javaee/6/api/javax/servlet/http/HttpServletRequest.html
18. ionic-forms-and-validations/src at master · ionicthemes/ionic-forms-and-validations. (2021). Retrieved 20 November 2021, from https://github.com/ionicthemes/ionic-forms-and-validations/tree/master/src
19. Framework, I. (2021). Ioinc 4 - form and submit button. Retrieved 20 November 2021, from https://forum.ionicframework.com/t/ioinc-4-form-and-submit-button/136988
20. AngularJS Integration with Java Servlet using MVC | Coding Boot (2021). Retrieved 20 November 2021, from https://www.youtube.com/watch?v=MRht3ZGjteA
21. java — cómo integrar Angular 2 + Java Aplicación web Maven. (2021). Retrieved 20 November 2021, from https://cutt.ly/zYD3iKz

22. How To Develop and Build Angular App With Java Backend. (2021). Retrieved 20 November 2021, from https://cutt.ly/nYD96FQ
23. International phone number? Read more how to format | CM.com. (2021). Retrieved 20 November 2021, from https://www.cm.com/blog/how-to-format-international-telephone-numbers/

15
Anexo

15.1. Futuras Mejoras

Como podemos ver la aplicación de Vestigial ha cumplido de forma correcta lo que en un principio propusimos.

Una de las posibles mejoras que podríamos plantear sería la posibilidad de almacenar la información de cada usuario en sus dispositivos permitiendo el trabajo de la aplicación sin necesidad de conexión a internet. Para ello lo que deberíamos de hacer sería implementar un nuevo componente en el front-end el cual se encargará de tanto almacenar en el dispositivo la información del usuario y que al iniciar la aplicación cuando posea conexión a internet compruebe el estado de los datos comparados en el servidor. Para ello lo que podríamos hacer fácilmente sería utilizar una función Hash la cual compruebe si los datos concuerdan con los almacenados del servidor.

De esta forma rápidamente podríamos comparar si los datos en el cliente son los mismos que los del servidor. Siendo los “elementos” que no concuerden los únicos que serán actualizados automáticamente.

Esto lo podríamos implementar de la siguiente forma, pero por falta de tiempo y por falta de conocimientos del funcionamiento nativo de los sistemas móviles no se ha podido desarrollar y por ello no se planteó en un principio.

Para llevar acabo esta tarea podríamos usar algoritmos de encriptación como MD5 o SHAE-256. Para así obtener los Hash de los datos y así fácilmente reconocer las modificaciones.

Otra posible mejora sería la posibilidad de aumentar los campos de los contactos y de los eventos permitiendo incluir campos como ubicación, contactos relacionados, etc.

Para ello se debería de realizar algunas mejoras tanto a la base de datos como a los DAO y DTO pero debido de que encapsulamos la información en los controladores podremos fácilmente realizar este cambio, debido a que en las interacciones entre el front-end al backend se realizan mediante json y esto permitiría fácilmente agregar los nuevos campos en los mensajes enviados.

Uno de los aspectos que se pretendía realizar pero no se sabía si daría tiempo sería el tema de permitir agregar fotos a los contactos y perfil del usuario esto último se empezó a desarrollar pero debido a los problemas que surgieron en la semana 6 del proyecto se dejaron parados. Para realizar esta mejora se debería de agregar un nuevo campo en las tablas de contactos en la base de datos del tipo **blob**, para así almacenar las imagenes, además con esto tendríamos que modificar en las vistas el como se visualiza los datos del contacto, lo cual no sería un cambio muy complejo. El problema reside a la hora de transferir la imagen del backend al front-end, debido a que no conseguimos averiguar el como transferir una imagen desde los controladores (desarrollados en Java) a las vistas (desarrolladas en Angular con Ionic).

Una vez visto esto, este aspecto nos hubiera permitido utilizar más integraciones del teléfono que nos harían distinguir más las diferencias entre un teléfono móvil y un ordenador ya que podríamos integrar la posibilidad de usar la cámara del móvil para agregar las fotos de perfil .