Informe Segundo Sprint

Raven

"Ayudarte a recordar"

Rubén Gabás Celimendiz (590738)
Eduardo Ibáñez Vásquez (528074)
Agustín Navarro Torres (587570)
Daniel Uroz Hinarejos (545338)

Grado en Ingeniería Informática
Escuela de Ingeniería y Arquitectura

20 de enero de 2016

Índice

1. Introducción

1.1. Descripción del proyecto

En este proyecto se ha optado por realizar una aplicación móvil para dispositivos Android que tenga valor para personas que sufran la enfermedad de Alzheimer. Para ello, se ha desarrollado el proyecto con la metodología ágil Scrum con una división en dos sprints de seis semanas cada uno (un total de 180 horas por sprint).

El proyecto cuenta con dos partes bien diferenciadas:

- La aplicación móvil que se entregará el cliente.
- El servidor y la base de datos para el correcto funcionamiento de la aplicación móvil.

La aplicación móvil se ejecutará en local en el dispositivo Android mientras que el servidor y la base de datos se encontrarán en máquinas remotas que se accederán a través de peticiones por Internet desde el dispositivo móvil.

1.2. Descripción del equipo

El equipo de desarrollo de Raven esta compuesto por cuatro desarrolladores con experiencia en distintas ámbitos: bases de datos, desarrollo de servidores, aplicaciones Android, administración de sistemas y control de versiones.

Los miembros del equipo son:

- Rubén Gabás Celimendiz: servidor, desarrollo Android y base de datos.
- Eduardo Ibáñez Vásquez: desarrollo Android y diseño.
- Agustín Navarro Torres: servidor, base de datos y administración de sistemas
- Daniel Uroz Hinarejos: desarrollo Android y base de datos.

2. Producto

2.1. Plan de producto

Raven es una aplicación móvil dirigida a personas con Alzheimer o problemas de memoria, con el fin de mejorar su día a día. Mejorando tanto su calidad de vida como la de sus familiares. Cuando se usa la aplicación, se está contribuyendo a mejorar el bienestar de estas personas.

Partiendo de esta premisa, mejorar la vida de la gente con Alzheimer, la aplicación permite al usuario poner eventos a una determinada fecha y hora, cronómetros, tener los números más utilizados o de emergencia a mano y jugar a juegos para mejorar su memoria y frenar el avance de la enfermedad. De esta manera los usuarios podrá tener una mayor autonomía y mejorar su calidad de vida.

Además la aplicación utiliza la nube para que familiares, amigos o cuidadores de la persona enferma puedan realizar un seguimiento de sus tareas y comprobar que han realizado acciones tales como tomar la medicación o comer, permitiendo tener un mayor seguimiento del enfermo y estando más tranquilos.

Además por último la aplicación guardará información relevante del usuario como nombre, problemas médicos o persona de contacto para que en caso de emergencia los profesionales puedan disponer de ella.

2.2. Planificación de lanzamientos

Lanzamiento	Objetivos	Duración
1º	Creación de eventos en el calendario Marcación rápida de llamadas Servidor Cronómetro y temporizador Cuentas de usuario Juego simón	1 año
2º	GPS Juego *sudoku*	6 meses
3º	Juego *memory* Diferenciación de roles	2 años

2.3. Análisis de riesgos

Se ha realizado un análisis de riesgos correspondiente al desarrollo del proyecto, y se han establecido a la vez una serie de contingencias:

Riesgos	**Probabilidad**	**Coste**	**Contigencia**
Pérdidad de información	Baja	Alto	Realización de copias de seguridad
Incapacidad de volver a una versión anterior	Baja	Alto	Utilización de un sistema de control de versiones
Alguna de las herramientas, por ejemplo librerías, no funcionan de la forma esperada	Alto	Medio	Estudio de las tecnologías previo a su utilización
Indisposición de alguno de los desarrolladores	Baja	Medio	El resto de desarrolladores conocerán el ámbito del trabajo del otro desarrollador y en qué se encuentra trabajando
Incompatibilidad entre sistemas, librerías, etc.	Alto	Bajo	Utilización de scripts que automaticen la compilación, gestión de dependencias, etc.
El tiempo planificado para cada sprint no coincide con el destinado	Medio	Medio	Intentar mejorar la planificación, añadir los PBI no realizados al inicio de la pila para su realización en los siguientes sprints
Fallos en el servicio de despliegue del servidor y base de datos	Bajo	Alto	Tener una máquina física donde poder desplegar el servidor y la base de datos en caso de fallo
Desconocimiento de las tecnologías por la mayor parte del equipo (NodeJS, API Calendario, MongoDB)	Medio	Alto	Apoyo de los integrantes del equipo que tienen experiencia en el desarrollo con estas tecnologías

2.4. Definición de hecho

Se considera que un hecho está completado cuando:

- Está correctamente documentado (el jefe de proyecto da el visto bueno)
- Está subido a GitHub rama master (se encuentra bajo control de versiones)
- Paso los test automatizados.
- Esta integrado con el resto de la aplicación desarrollada hasta ese momento.
- Cumple los requisitos funcionales y no funcionales asociados a ellos, el dueño de producto lo da por bueno.
- El dueño de producto da la vista buena al diseño de la aplicación.

2.5. Pila del producto en el estado actual

Todos los PBI planeados para este sprint han sido completados, por lo que la pila de producto tras la realización del segundo sprint ha quedado de la siguiente forma:

Prioridad	PBI	Tamaño	Completado
Primer sprint			
1	Crear usuario cuenta	M	Sí
2	Log in/out	S	Sí
3	Crear calendario	M	Sí
4	Crear evento calendario	S	Sí
5	Visualizar eventos calendario	M	Sí
6	Visualizar evento	M	Sí
7	Visualizar usuario cuenta	S	Sí
Segundo sprint			
8	Juego *simon*	L	Sí
9	Temporizador	L	Sí
10	Borrar evento calendario	M	Sí
11	Borrar usuario cuenta	S	Sí
12	Modificar evento calendario	M	Sí
13	Modificar usuario cuenta	S	Sí
Por desarrollar			
14	Agregar contactos	M	No
15	Visualizar marcación rápida	S	No
16	Visualizar contactos	S	No
17	Borrar contactos	M	No
18	Modificar contactos	M	No
19	Diferenciación de roles	[]	No
20	Juego *memory*	[]	No
21	Juego *sudoku*	[]	No
22	Localización GPS	XL	No

2.6. Historia de usuario

Crear una cuenta de usuario:

El usuario de la aplicación será capaz de crear una cuenta de usuario desde la aplicación Android y posteriormente podrá logearse con ella.

Log in/log out:

El usuario de la aplicación será capaz de entrar con su cuenta de usuario en la aplicación Android y desde acceder a la información de su cuenta. De la misma forma un usuario podrá ser capaz de salir de su cuenta de usuario desde Android, sin poder ver la

información asociada a su usuario desde ese momento y volviendo a la pantalla de login.

Crear calendario:

El usuario de la aplicación será capaz de visualizar un calendario dentro de la aplicación Android.

Crear eventos calendario:

El usuario de la aplicación será capaz de crear un evento en el calendario asociado a un día.

Visualizar eventos calendarios:

El usuario de la aplicación será capaz de ver los días en los cuales existen eventos diferenciados de los días en los cuales no existe eventos.

Visualizar evento:

El usuario de la aplicación será capaz de ver con detalle los eventos de un día seleccionado.

Visualizar la cuenta de usuario:

El usuario de la aplicación será capaz de crear una cuenta de usuario desde la aplicación Android y posteriormente podrá logearse con ella.

Juego simon:

El usuario de la aplicación será capaz de jugar al simon dentro de ella. El simón es un juego que genera una secuencia de colores y sonidos que el usuario tiene que repetir.

Temporizador:

El usuario será capaz de programar un temporizador con un título el cual tras el tiempo previamente indicado por el usuario saltará una alerta.

Borrar evento del calendario:

El usuario será capaz de borrar un usuario de su calendario previamente creado.

Borrar cuenta de usuario:

El usuario podrá borrar su cuenta de usuario y toda la información asociado a ella.

Modificar evento del calendario:

El usuario será capaz de modificar la información de un usuario previamente creado.

Modificar cuenta de usuario:

El usuario será capaz de modificar la información asociada a su cuenta de usuario.

2.7. Criterios de aceptación

Los criterios de aceptación para los PBI desarrollados durante los dos sprints son los siguientes:

Crear usuario cuenta:

- Se dispone de una GUI, en la aplicación Android, diseñada para la creación de cuenta de usuario.
- La cuenta de usuario tendrá los siguientes campos de usuario: número de teléfono, email, contraseña, nombre, apellido, información del usuario, lugar de residencia, fecha de nacimiento, nombre de una persona de contacto, apellido de una persona de contacto, teléfono de una persona de contacto.
- Se cuenta con un documento en la base de datos que refleja todos los campos asociados a una cuenta de usuario.
- En caso de que el usuario introduzca datos erróneos o tenga campos vacíos, se le mostrar al usuario un aviso indicando que los datos introducidos son erróneos, este mensaje tiene carácter general.

Log in / log out:

- Se dispone de una GUI, en la aplicación Android, para que el usuario pueda logearse.
- Un usuario se podrá logear dentro de la aplicación, si la cuenta existe previamente en la base de datos de la aplicación, con su email y contraseña.
- En caso de que exista un problema y el usuario no pueda logearse con su cuenta de usuario, el email y/o contraseña son incorrectos, la cuenta de usuario no existe, etc. Se mostrara un aviso, un pop-up, al usuario indicando que no se ha podido realizar el login. Este aviso es general y no entrará en detalles de por que no se ha podido realizar correctamente el login.
- Existe un opción en la pantalla principal de aplicación, aquella que aparece nada más realizar login en la aplicación (ver bocetos de la aplicación), que permite al usuario hacer log out de la aplicación Android.

Crear calendario:

- Se dispone una GUI en la aplicación Android que muestra un calendario. Este calendario muestra todos los días de un mes, organizados por semanas. Los días que presenten algún evento tendrán un punto debajo suyo. Los eventos de cada día aparecen en una lista.

Crear evento calendario:

- Se dispone de una GUI en la aplicación Android en la que el usuario podrá crear un evento. Un evento dispondrá de los siguientes campos: Mensaje del evento, fecha del evento, hora del evento y un checo para cada día de la semana.
- El usuario podrá especificar que días de la semana se repetirá ese evento.

Visualizar evento calendario:

- Los días en los cuales hay un evento están reflejados en el calendario mediante un punto en el día.
- Cuando se esta seleccionando un día que tiene un evento o más, estos aparecen en la parte inferior de la pantalla en forma de listado con el mensaje y la hora en la cual esta configurado el evento.

Visualizar evento:

- Se dispone de una GUI en la cual se puede visionar con detalle la información de un evento: el mensaje, fecha, hora y días en los que se repite.
- Se puede acceder a la visualización de un evento concreto, a la pantalla de la GUI, mediante la selección del evento desde el calendario con el día del evento seleccionado.

Visualizar usuario cuenta:

- Se dispone de una pantalla en la GUI en la cual se puede visionar con detalle toda la información de un usuario: nombre, apellido, email, año de nacimiento, teléfono, información médica, lugar de residencia, contraseña, y la información de un familiar - nombre del contacto, apellido del contacto, teléfono del contacto -.
- Se dispone de un botón desplegable en el menú principal que permite acceder a la pantalla de la GUI para la visualización de la información del usuario cuenta.

Juego simon:

- Se dispone de una GUI, como la vista en la figura x, en la aplicación Android.
- El juego simon cuenta con cuatro botones de distintos colores – rojo, Amarillo, verde y azul – y colores.
- La dificultad del juego es incremental según el usuario va respondiendo correctamente al juego.

Temporizador

- Se dispone de una GUI, como la vista en la figura x, en la aplicación Android.
- El temporizador tendrá un titulo personalizable por el usuario.
- El usuario podrá indicar las horas, minutos y segundos en los cuales saltará la alarma.
- La alarma del temporizador funcionara con la aplicación en segundo plano.
- Cuenta con notificaciones que muestra las alarmas activas actualmente.

Borrar eventos calendarios:

- Se dispone de una GUI, como la vista en la figura x, en la aplicación Android.
- El usuario será capaz de borrar un usuario previamente creado por el mismo en el calendario.

Modificar evento calendario:

- Se dispone de una GUI, como la vista en la figura x, en la aplicación Android.
- El usuario será capaz de modificar la información asociada a un evento previamente creado por el usuario en el calendario.
- La información que podrá modificar el usuario del evento será la siguiente: mensaje del evento, fecha del evento, hora del evento y días en los cuales se repetirá el evento.

Modificar un usuario cuenta:

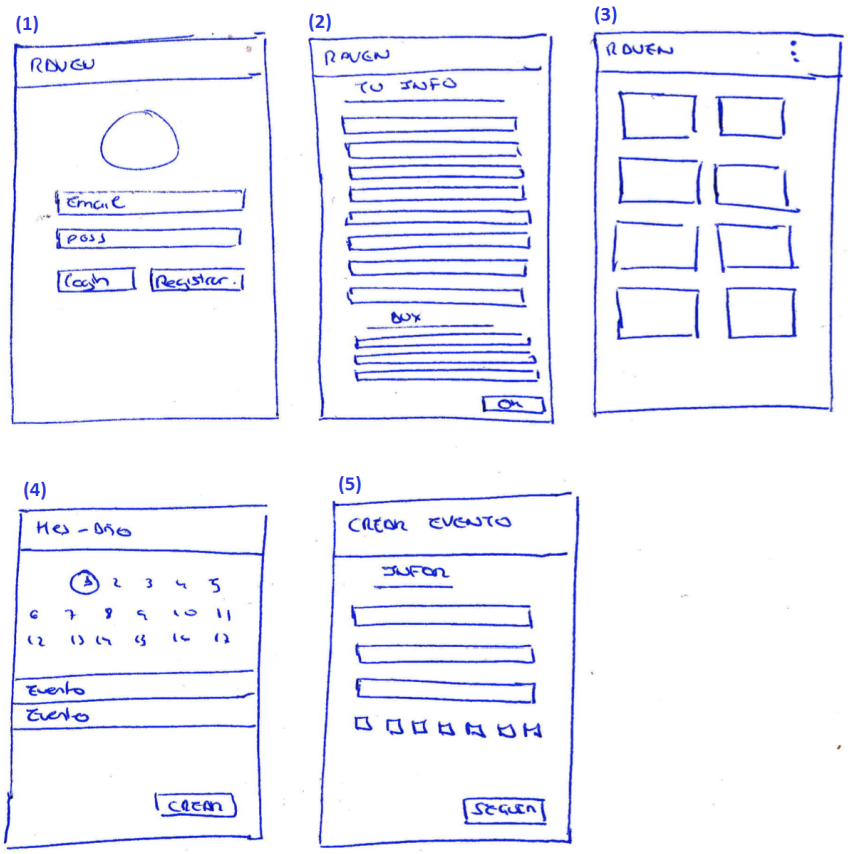
- Se dispone de una GUI, como la vista en la figura x, en la aplicación Android.
- El usuario es capaz de modificar la información asociada a un evento previamente creado por el usuario en el calendario.
- La información que se podrá modificar del usuario será la siguiente: número de teléfono, email, contraseña, nombre, apellido, información del usuario, lugar de residencia, fecha de nacimiento, nombre de una persona de contacto, apellido de una persona de contacto, teléfono de una persona de contacto.

Borrar un usuario cuenta:

- Se dispone de una GUI, como la vista en la figura x, en la aplicación Android.
- El usuario será capaz de borrar su cuenta de usuario y los eventos asociados a ella.
- Una vez borrada la cuenta, el usuario no podrá realizar ninguna acción con ella.

2.8. Bocetos de la GUI

Se han realizado los siguientes bocetos de la GUI para realizar la GUI correspondiente al **primer sprint**.



(1) Pantalla de inicio de la aplicación. Relacionada con el PBI "Log in/out"

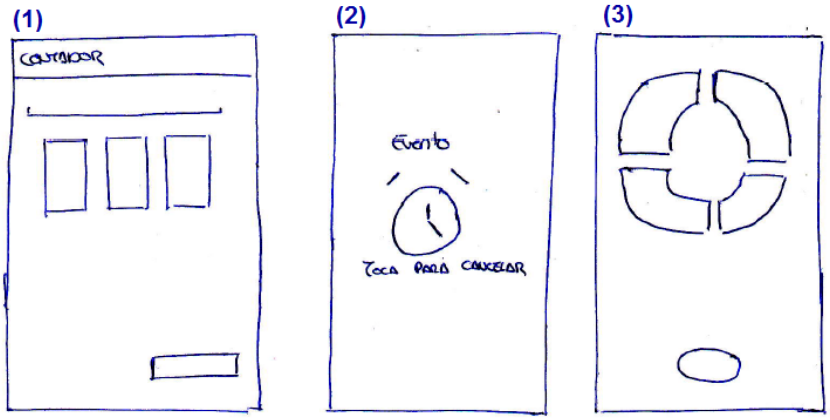
(2) Pantalla de registro de la aplicación. Relacionada con el PBI "Crear usuario cuenta" y "Visualizar usuario cuenta"

(3) Pantalla del menú principal de la aplicación.

(4) Pantalla de presentación del calendario con sus eventos. Relacionada con el PBI "Crear calendario" y "Visualizar eventos calendario"

(5) Pantalla de creación de evento de la aplicación. Relacionada con el PBI "Crear evento calendario" y "Visualizar evento"

Se han realizado los siguientes bocetos de la GUI para realizar la GUI correspondiente al **segundo sprint**.



(1) Pantalla de creación de un contador. Relacionada con el PBI “Temporizador”

(2) Pantalla de alarma de un temporizador. Relacionada con el PBI “Temporizador”

(3) Pantalla del juego simon. Relacionada con el PBI “Juego simon”

2.9. Estado actual de la aplicación

2.9.1. Funcionalidad implementada

Actualmente se ha implementado las siguientes funcionalidades a la aplicación:

- Creación de usuarios.
- Log in/out del usuario.
- Visualización de la información del usuario.
- Modificar la información asociada a una cuenta de usuario.
- Borrar la información asociada a una cuenta de usuario.
- Visualización (creación) de un calendario.
- Creación de eventos de un calendario.
- Visualización de eventos en el calendario.
- Visualización de eventos en detalles del calendario.
- Modificación de la información de un calendario.
- Borrar eventos de un calendario.
- Configuración de un temporizador con un mensaje.
- Juego simon.

2.9.2. GUI implementada hasta el momento

Se adjunta las imágenes de la GUI implementada hasta al momento en la aplicación:

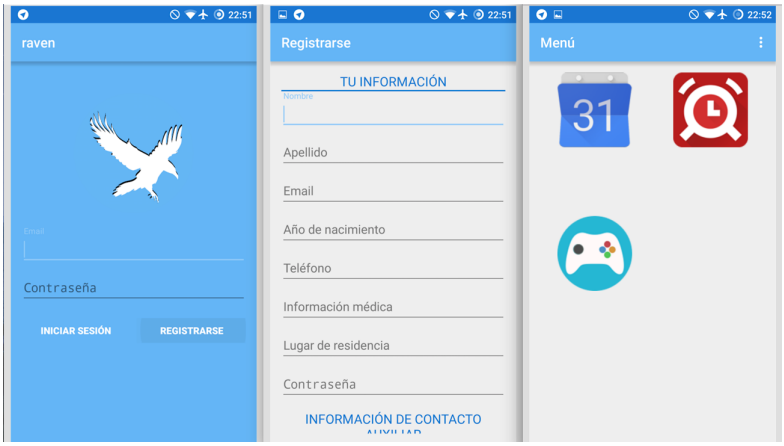


Figura 1: (De izquierda a derecha) Pantalla de inicio, pantalla de registro, pantalla de menú.

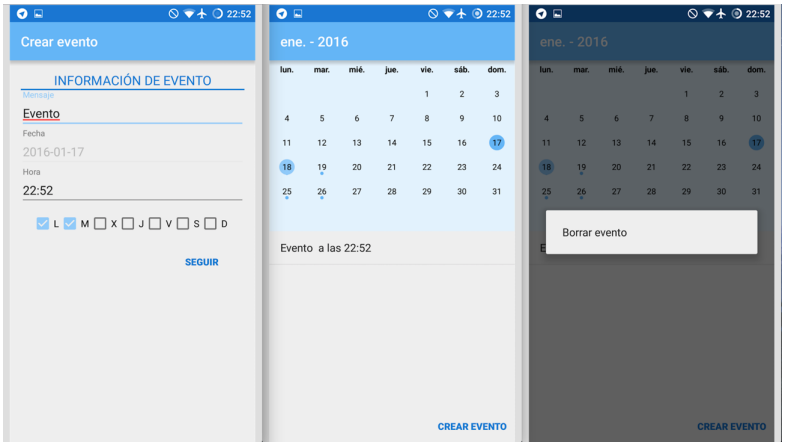


Figura 2: (De izquierda a derecha) Creación/visualización/modificación de evento, visualización de calendario, botón para borrar un evento.

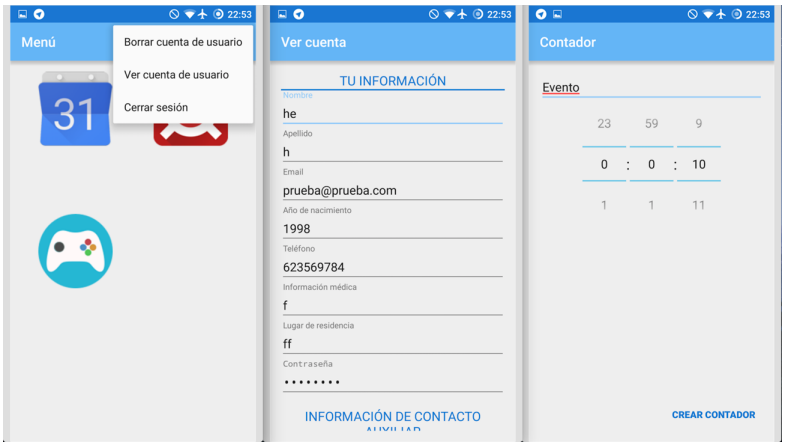


Figura 3: (De izquierda a derecha) Botón con las opciones de usuario, pantalla para la modificación/visión de la información del usuario, pantalla de creación de un temporizador.

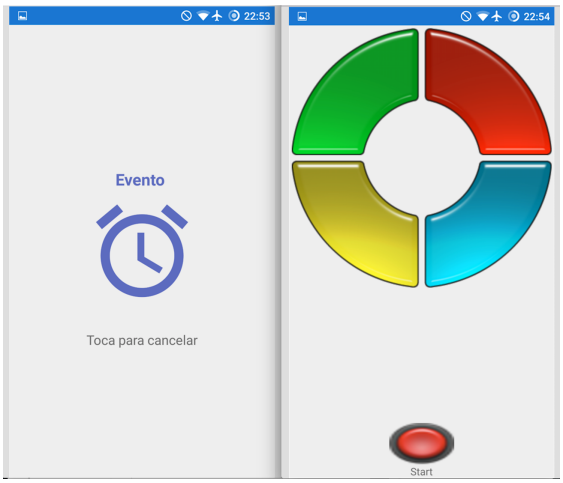
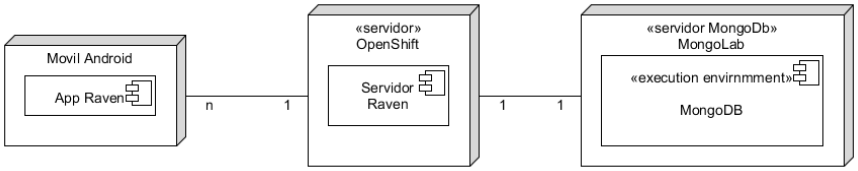


Figura 4: (De izquierda a derecha) Pantalla de notificación de evento mediante el temporizador, juego simon.

La GUI implementada corresponde a la funcionalidad implementada, vista en el apartado anterior.

2.10. Arquitectura y diseño

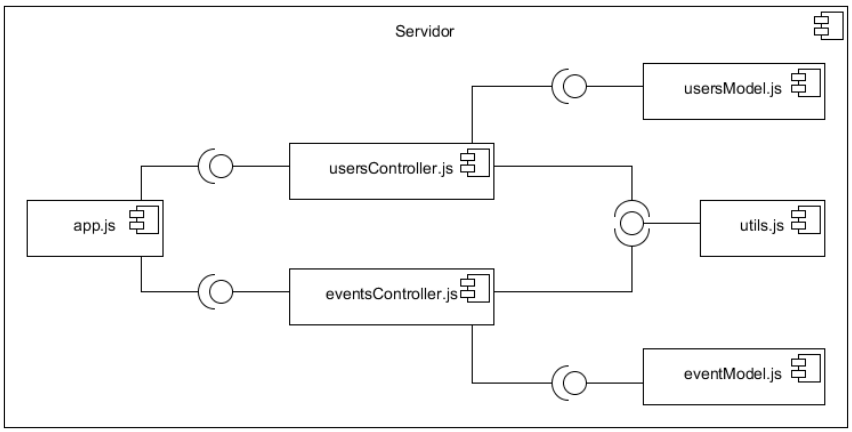
2.10.1. Despliegue



Se ha optado por la siguiente arquitectura reflejada en el diagrama de despliegue debido a que presenta los siguientes puntos fuertes:

- Se trata de una aplicación que sigue una estructura de cliente-servidor. La interfaz no tiene estado, dado que las peticiones que se realizan entre cliente-servidor son HTTP siguiendo el estilo arquitectural REST. La comunicación entre ambas máquinas se realiza de manera asíncrona.
- Se ha separado el servidor y la base de datos en dos máquinas diferentes con el fin de conseguir una mayor escalabilidad, dado que esta distribución permite administrar la carga de trabajo entre varios servidores distintos en caso de necesidad.
- El tener un único servidor permite la centralización de los recursos, y que en una primera instancia, consiguiendo que la integración sea sencilla. Sin embargo, como se ha comentado anteriormente, gracias a la arquitectura que sigue el back-end de la aplicación, podrían añadirse nuevos servidores si hiciera falta. De esta manera se conseguiría una mayor tolerancia a fallos, dado que la caída de una de ellas no significaría que la aplicación dejara de prestar el servicio, dado que el resto de servidores de la aplicación se encargarían de proporcionarlo.
- Facilidad de mantenimiento e integración de todos los servicios. Todos estos servicios se solicitan mediante peticiones HTTP en REST.
- Con el fin de mejorar más la escalabilidad de la aplicación, tiene la posibilidad de aumentar también el número de nodos MongoDB, realizando un escalado horizontal, y de este modo repartir la carga de trabajo entre varias máquinas.
- Es un estructura conocida por todos los desarrolladores de la aplicación.

2.10.2. Componentes del servidor



Se ha optado por esta distribución de componentes reflejada en el diagrama debido a las siguientes razones:

- La configuración de la aplicación se mantiene en único componente - app.js - de manera que sufra los menores cambios posibles a lo largo del desarrollo, y en caso de que los sufra, sea fácil su modificación y no se vea afectado por el resto de la aplicación.
- Los controladores están separados según su naturaleza, usuario o eventos en el primer sprint, de manera que cada uno de los elementos tenga acotados su rango de acción y evitar el desarrollo de grandes componentes que aglutinan muchas servicios, dificultando el mantenimiento del código.
- Escalabilidad: puesto que cada controlador se puede poner en distintos servidores sin que suponga ningún problema.
- La conexión con la base de datos se hace a través de eventsModel.js y usersModel.js, de manera que se consigue desacoplar la interfaz y lógica de la aplicación - controladores - y la conexión con la base de datos.
- Aquellos métodos o elementos que podrían utilizarse entre varios elementos del servidor - controladores o modelos - están incluidos en útil.
- Se ha intentado conseguir que los componentes estén lo más desacoplado entre ellos, de manera a que la modificación de un elemento, siempre y cuando cumpla especificación y funcione correctamente, no afecte al resto de elementos.
- Se ha intentado conseguir una aplicación mantenible, con capacidad para escalar, facilidad de despliegue y la menor deuda técnica posible.

- Permite mantener la aplicación sencilla facilitando su futuro desarrollado, y que nuevos desarrolladores se puedan unir posteriormente al proyecto sin grandes complicaciones.

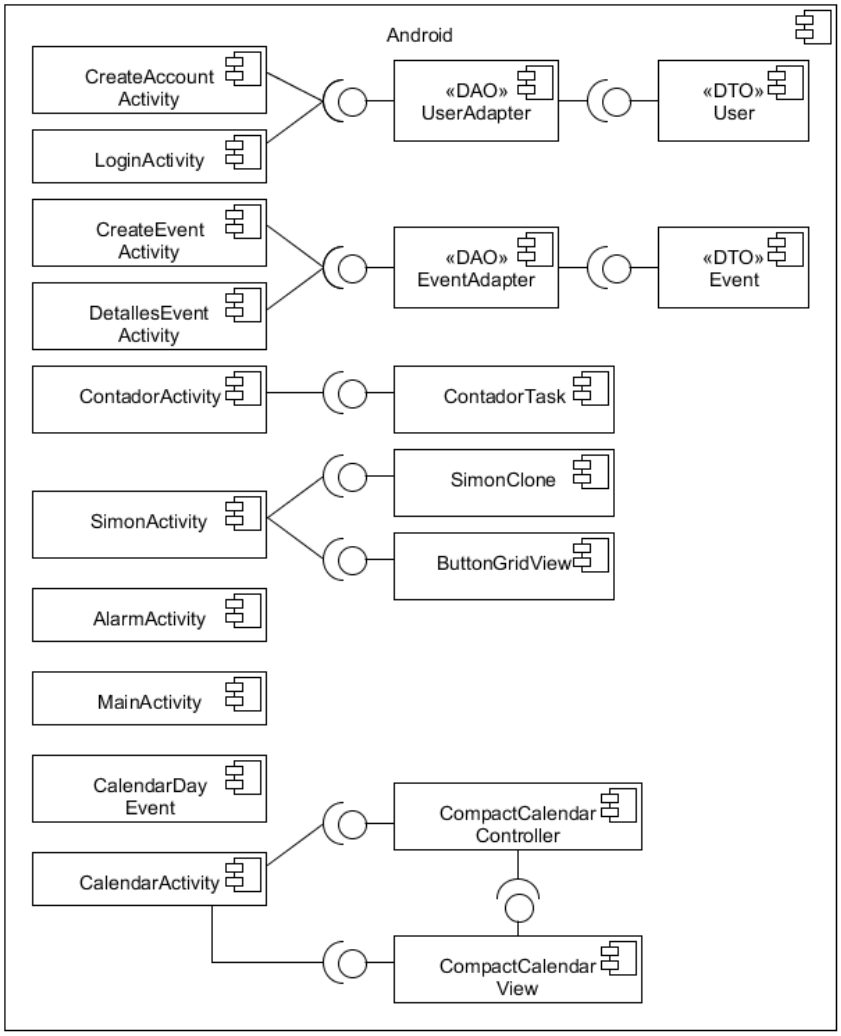
Interfaces de usersController

- /**createUser**: recibe los datos de un usuario en formato JSON, añadiéndolos en la base de datos de la aplicación. Devuelve un código de estado HTTP, dependiendo de si la operación se ha realizado correctamente o no.
- /**loginUser**: recibe el email y la contraseña de un usuario en formato JSON, para comprobar que coinciden con los almacenados en la base de datos. En caso que los datos sean válidos, devolverá un código de estado HTTP 200, mientras que enviará un 400 en caso contrario.
- /**findUser**: recibe el email de un usuario y devuelve los datos del usuario a los que corresponda ese email en la base de datos de la aplicación. En caso de error, devuelve un código de estado 400.
- /**modifyUser**: recibe el email de un usuario, así como el resto de datos que se tienen de ese usuario en formato JSON y devuelve los datos del usuario una vez se hayan modificado en la base de datos. En caso de error, devuelve un código de estado 400.
- /**deleteUser**: recibe el email de un usuario en formato JSON. Devuelve un 200 en caso de que el usuario con el email obtenido se haya eliminado de la base de datos y un 400 si la operación no ha sido realizada con éxito.

Interfaces de eventsController

- /**createEvent**: recibe los datos de los que se compone un evento en formato JSON y lo añade a la base de datos de la aplicación. Devuelve un 200 si se ha realizado todo con éxito, un 400 en caso contrario.
- /**getEvents**: recibe el email de un usuario de la aplicación y devuelve en forma de lista los eventos asociados a ese usuario. En caso de fallo devuelve un código 400.
- /**getEvent**: recibe el id de un evento y devuelve los datos del evento al que corresponda ese id. En caso de error devuelve un 400.
- /**modifyEvent**: recibe el id de un evento, así como todos los datos que forman un evento y modifica en la base de datos los valores del evento que tenga el id recibido. Devuelve los datos del evento modificado en caso de éxito y un código de esta-do 400 en caso contrario.
- /**deleteEvent**: recibe el id de un evento en formato JSON y elimina de la base de datos el evento que posea ese id. En caso de éxito devuelve un 200 y en caso de fracaso, un 400.

2.10.3. Componentes de la aplicación Android



Se ha optado por ese diagrama de componentes debido a las siguientes razones:

- Porque así se tiene separadas las vistas (Parte de los Activities), la lógica (Parte de los Activities, CompactCalendarController, UserAdapter y EventAdapter) y el modelo de la aplicación (Event y User). Aún así, existe cierto acoplamiento entre la lógica y la vista debido a la propia naturaleza de Android.
- Gracias a la división en MVC cada uno de los elementos se encarga de una funcionalidad concreta y evita grandes componentes que aglutinan muchos servicios.

Además permite tener distintas vistas para representar una determinada información en la propia aplicación.

- Las clases UserAdapter y EventAdapter son las encargadas de realizar las peticiones HTTP al servidor de la aplicación. Estas peticiones siguen el estilo de arquitectura REST, son asíncronas y llevan los datos al servidor codifica-dos en formato JSON, que es un estándar para el intercambio de datos.

3. Proceso

3.1. Test

3.1.1. Servidor

Se ha decido que para cada una de las funcionalidades implementadas en el servidor, relacionados a un PBI, se compruebe su correcto funcionamiento mediante un test que compruebe su correcta implementación, con el fin de conseguir que una amplia cobertura del código mediante test y un servidor robusto.

Para la realización de los test en el servidor se han utilizado las siguientes herramientas:

- Mocha + Chai: como suit para la realización de los test.
- Istanbul: para medir la cobertura de test.
- Travis: para comprobar que las distintas ramas del proyecto pasa correctamente los test.
- Codecov: para medir la cobertura de los test dentro del proyecto.

Todos los test realizados en el servidor son de tipo unitario.

3.1.2. Cliente

En el cliente se han realizado los test unitarios sobre las operaciones de datos de usuarios, y los de eventos ya que son los datos fundamentales para que funcione correctamente la aplicación. Muchas de las partes que no se contemplan en los test unitarios son las relacionadas con funciones propias de android o que su mero funcionamiento se prueban en las pruebas de aceptación como pueden ser temas relacionados con la GUI.

Herramientas utilizadas para la realización de los test de esta parte son:

- Junit: para realizar los test unitarios.
- Robotium: para realizar las pruebas de aceptación.
- Travis: para comprobar que las distintas ramas del proyecto pasa correctamente los test.
- Codecov: para medir la cobertura de los test dentro del proyecto.

3.2. Control de versiones

Para el control de versiones se ha utilizado Git y GitHub, en el cual se ha optado por la creación de una organización con dos repositorios:

- Raven: contiene la aplicación Android.
- Servidor: contiene el servidor de la aplicación.

Así pues la manera en la cual se a trabajado contra GitHub durante la realización del proyecto ha sido la siguiente: todos los usuarios clonaran los repositorios en local y trabajaran sobre ellos directamente, asegurando de que los cambios realizados sobre el mismo pasa los distintos test existentes en el servidor. En caso de que se vaya a realizar algún cambio mayor que pueda afectar al funcionamiento del resto de la aplicación, por ejemplo cambiar los colores de la aplicación o cambiar la manera con la cual se conecta al servidor, se creara un nueva rama y cuando se hayan realizado todos los cambios se realizará un merge con la master. La documentación del proyecto se ha decido que se encuentre en la wiki del servidor.

3.3. Diagramas de trabajo completado

Como se puede observar en la gráfica, la realización completa de los PBI varía a lo largo de la vida del primer sprint. Inicialmente no se llegan a completar los PBI debido a que la puesta en marcha del proyecto es la tarea más compleja.

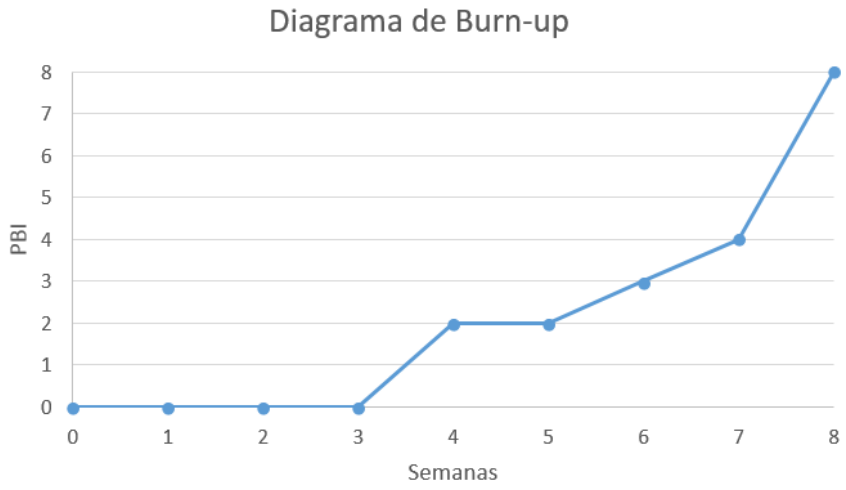


Figura 5: Primer sprint

Una vez realizada toda la infraestructura del proyecto se puede proceder a la realización de los PBI a lo largo del sprint. En contrapartida, se espera que en el siguiente sprint la realización de los PBI ocurran desde el inicio, al igual que una realización de PBI más uniforme a lo largo del tiempo.

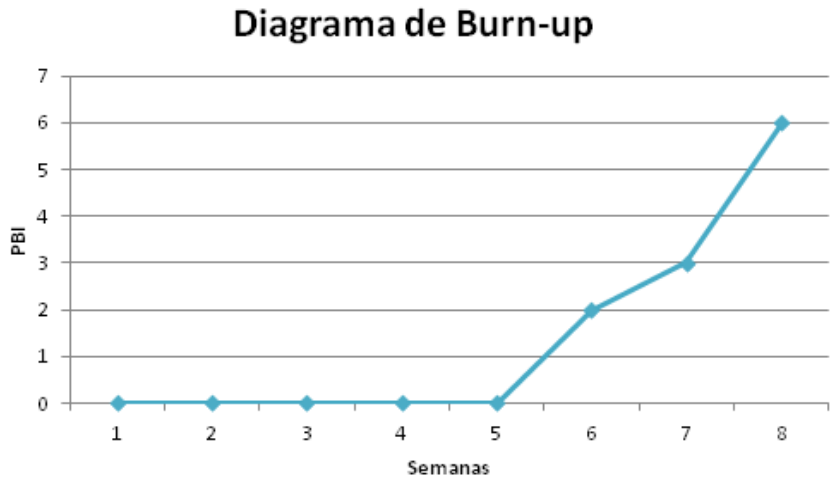


Figura 6: Segundo sprint

3.4. Esfuerzos por actividad

Primer sprint		
PBI	**Tamaño**	**Tareas**
Crear cuenta de usuario	M (8)	Protocolo de comunicación
		Configuración del servidor.
		Crear base de datos.
		Lógica y GUI de la app Android.
		Tests automatizados.
		Documentación.
Log in/out	S (2)	GUI y lógica de la app Android.
		Configuración del servidor.
		Lógica del servidor.
		Test automatizados
		Protocolo de comunicación.
		Documentación.
Crear calendario	M (3)	Investigación.
		GUI
		Test
		Documentación.
Crear evento calendario	M (5)	Base de datos.
		Protocolo de comunicación.
		GUI y lógica de la app Android.
		Test automatizados.
		Documentación.
Visualizar eventos calendario	M (1)	GUI y lógica del cliente.
		Lógica del servidor.
		Sincronización.
		Test automatizados.
		Protocolo de comunicación.
		Documentación.
Visualizar evento	M (1)	GUI y lógica del cliente.
		Lógica del servidor.
		Test automatizados.
		Protocolo de comunicación.
		Documentación
Visualización del usuario cuenta	S (1)	GUI y lógica de la aplicación.
		Lógica del servidor.
		Test automatizados.
		Protocolo de comunicación.
		Documentación.

Segundo sprint		
PBI	**Tamaño**	**Tareas**
Juego simon	L (8)	Diseño
		Lógica
		Test
		Documentación
Temporizador	L (5)	Diseño
		Lógica
		Test
		Documentación
Borrar evento calendario	M (2)	Lógica del cliente
		Lógica del servidor
		Test
		Documentación.
Borrar usuario cuenta	S (1)	Lógica del cliente
		Lógica del servidor
		Test
		Documentación.
Modificar evento calendario	S (1)	Lógica del cliente
		Lógica del servidor
		Test
		Documentación.
Modificar usuario cuenta	S (1)	Lógica del cliente
		Lógica del servidor
		Test
		Documentación.

3.5. Esfuerzos por persona

El esfuerzo realizado en el **primer sprint** por todo el equipo de desarrollo ha sido de 141 horas, por lo que se han dedicado 28 horas más de las previstas para el desarrollo del primer sprint. Además, existen una horas 122,5 horas que no se contabilizan como desarrollo del proyecto que forman parte de horas de formación (clases incluidas).

A continuación el desglose de horas por miembro del equipo que ha dedicado al proyecto:

- Daniel Uroz Hinarejos: 22,5
- Eduardo Ibáñez Vásquez: 39,5
- Rubén Gabás Celimendiz: 41
- Agustín Navarro Torres: 38

Primer sprint					
PBI	**Tareas**	**Daniel**	**Edu**	**Rubén**	**Agustín**
Crear cuenta de usuario	Protocolo de comunicación.	0	4	0	2
	Configuración del servidor.	0	0	3	3
	Crear base de datos.	0	0	2	1
	Lógica y GUI de la app Android.	0	4	0	0
	Tests automatizados.	0	4	3	1
	Documentación.	0	1	0	2
Log-in/out	GUI y lógica de la app Android.	0	3	0	0
	Lógica del servidor.	0	0	3	2
	Test automatizados	0	1	2	2
	Protocolo de comunicación.	0	1	0	1
	Documentación.	0	1	0	1
Crear calendario	Investigación.	2	0	2	0
	GUI	4	0	0	0
	Test	1	0	2	0
	Documentación.	1	0	0	0
Crear evento calendario	Base de datos.	0	0	2	1
	Protocolo de comunicación.	1	1	0	1
	GUI y lógica de la app Android.	1	4	0	0
	Test automatizados.	0	2	1	0
	Documentación.	0	0	0	2
Visualizar eventos calendario	GUI y lógica del cliente.	4	0	2	0
	Lógica del servidor.	2	0	3	2
	Sincronización.	2	0	2	1
	Test automatizados.	0	0	2	2
	Protocolo de comunicación.	1	0	2	1
	Documentación.	0	0	0	1
Visualizar evento	GUI y lógica del cliente.	2	5	2	0
	Lógica del servidor.	1.5	0	3	3
	Test automatizados.	0	2	1	1
	Protocolo de comunicación.	0	1	0	1
	Documentación	0	0	0	1
Visualización del usuario cuenta	GUI y lógica del cliente.	0	0	0	1
	Lógica del servidor.	0	2.5	0	0
	Test automatizados.	0	2	1	1
	Protocolo de comunicación.	0	1	1	1
	Documentación	0	0	0	1

El esfuerzo estimado para todo el equipo de desarrollo en el **segundo sprint** es de 143 horas, mientras que el esfuerzo efectuado ha sido de 116 horas, por lo que se han necesitado invertir 27 horas menos de las previstas para realizar las tareas programadas

para este sprint. Además, el equipo de desarrollo ha empleado unas 32,5 horas que no se contabilizan como desarrollo del proyecto para formación (clases incluidas).

A continuación el desglose de horas por miembro del equipo que ha dedicado al proyecto:

- Daniel Uroz Hinarejos: 38
- Eduardo Ibáñez Vásquez: 34
- Rubén Gabás Celimendiz: 34
- Agustín Navarro Torres: 39

Segundo sprint					
PBI	**Tareas**	**Daniel**	**Edu**	**Rubén**	**Agustín**
Juego simon	Diseño	0	0	8	0
	Lógica	0	0	10	0
	Test	0	0	4	0
	Documentación	0	0	5	3
Temporizador	Diseño	5	0	0	0
	Lógica	6	0	0	0
	Test	4	0	0	0
	Documentación	5	0	3	3
Borrar evento calendario	Lógica de cliente	0	3	0	0
	Lógica del servidor	0	0	0	3
	Test	2	1	0	3
	Documentación	3	4	1	3
Borrar usuario cuenta	Lógica de cliente	0	3	0	0
	Lógica del servidor	0	0	0	3
	Test	2	1	0	3
	Documentación	3	4	1	2
Modificar evento calendario	Lógica de cliente	0	5	0	0
	Lógica del servidor	0	0	0	3
	Test	2	1	0	3
	Documentación	2	4	1	2
Modificar usuario cliente	Lógica de cliente	0	3	0	0
	Lógica del servidor	0	0	0	3
	Test	2	1	0	3
	Documentación	2	4	1	2

3.6. Otros aspectos de la gestión del proyecto

Otras herramientas utilizadas para la gestión del proyecto y su desarrollo han sido las siguientes:

- Trello: para la organización del tablero del sprint.
- WhatsApp: para la comunicación dentro del equipo.
- Android Studio: para el desarrollo del cliente.
- NPM y Gradle: como scripts para facilitar a compilación, gestión de dependencias, lanzamiento de test, etc.
- OpenShift: para desplegar el servidor en la nube.
- MongoLab: como base de datos.

3.7. Medidas para mejorar el proyectos

En la última retrospectiva se establecieron una serie de medidas con el fin de mejorar la gestión y desarrollo del proyecto en los próximos sprints:

- Automatización de pruebas de aceptación.
- Despliegue automático en OpenShift.
- Mantenimiento actualizado de tablas de sprint.
- Diagramas burn down/up durante el desarrollo del sprint.
- Mejora de la comunicación entre el equipo de desarrollo.
- Tener más presente el estado de la pila del producto en Trello durante el desarrollo.

4. Conclusiones

En el documento se ha intentado exponer de la manera más concisa los puntos más relevante de la aplicación desarrollada durante los sprints, destacando:

- El diseño arquitectónico de la aplicación y por qué se ha llegado a él.
- La pila de producto, los PBI realizados con su definición de hecho, criterios de aceptación etc.
- Una breve descripción del proceso con el cual se ha realizado el proyecto.

Durante ambos sprints se consiguió realizar todos los PBI planificados, con una inversión ligeramente inferior de tiempo al planificado, todo ellos respaldados por test y preparados para funcionar.

Destacar que se buscó y se llegó al nivel de sobresaliente durante la realización del mismo, cumpliendo los siguientes requisitos:

1. El código del proyecto se encuentra alojado en GitHub. Repositorio de la aplicación: `https://github.com/UNIZAR-30248-2015-Raven`
2. La cobertura de test alcanza el 30 % del código, se puede observar la cobertura del servidor en codecov (`https://codecov.io/github/UNIZAR-30248-2015-Raven/Servidor`) y la del cliente en los test desarollados.
3. Definición de hecho clara, ver apartado 2.4.
4. Documentación arquitectural adecuada y justificada, ver apartado 2.10.
5. Cumplimiento suficiente de los requisitos.
6. Se ha llevado acabo las hojas de esfuerzos, siendo entregadas periódicamente.
7. La compilación y gestión de dependencias está basada en scripts: se puede observar gradle en la aplicación Android y npm en el servidor.
8. Toda la documentación del proyecto se encuentra bajo control de versiones: control de versiones de Dropbox y GitHub.
9. Todas las historias de usuario tienen criterios de aceptación validos: ver apartado 2.7.
10. Hay test de aceptación automáticos para al menos un criterio de aceptación: en el código se puede observar la existencia de un test de aceptación automático.
11. La compilación, ejecución de test y lanzamiento del servidor se encuentra automatizada bajo un script. Además el servidor se despliega de forma automática en OpenShift.

A. Manual de usuario

Título	Raven Manual de usuario
Versión	1.2
Realizado	OPdevelopers
Fecha	20/01/2016

Control de versiones		
Versión	**Descripción**	**Fecha**
1.0	Documento inicial	18/01/2016
1.1	Secciones añadidas	19/01/2016
1.2	Todas las secciones añadidas	20/01/2016

A.1. Objeto del documento

El objeto del documento de manual de usuario pretende mostrar al usuario el funcionamiento de la aplicación Raven y las opciones que ofrece la misma.

A.2. Manual de usuario

A.2.1. Pantalla inicial

La pantalla inicial es la primera pantalla de la aplicación y nos ofrece 2 posibles opciones:

a) Si no se posee un usuario dado de alta en el sistema, pulsar botón "REGISTRARSE", este botón abrirá una ventana que veremos más adelante con los campos necesarios para poder darse de alta en la aplicación

b) Si se posee un usuario dado de alta en el sistema, se podrá rellenar los campos correspondientes al "EMAIL" y "CONTRASEÑA" y pulsar el botón "INICIAR SESIÓN", si los datos son correctos se accederá a la aplicación, si son incorrectos mostrará un mensaje de error.

A.2.2. Pantalla de registro

La pantalla de registro ofrece una serie de campos a rellenar: Nombre, apellido, email, año de nacimiento, teléfono, información médica, lugar de residencia y contraseña.

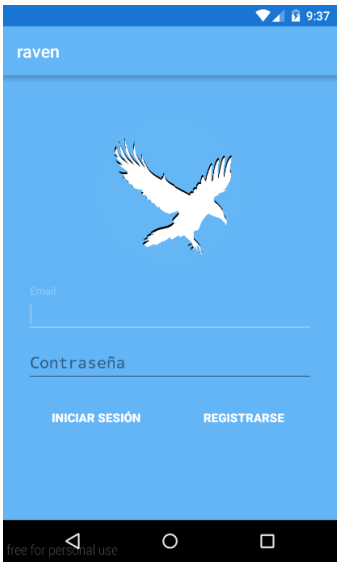


Figura 7: Pantalla inicial

Y además datos de contacto auxiliar: Nombre del contacto, apellido del contacto, teléfono del contacto.

Todos los campos del registro son obligatorios.

Una vez se rellenen todos los campos pulsar sobre el botón “ACEPTAR”, si no hay ningún error en los datos introducidos, se habrá dado de alta en la aplicación correctamente, si hay algún error en los datos se mostrará en cual campo está el error.

A.2.3. Pantalla menú

La pantalla de menú es la pantalla inicial una vez logueado en la aplicación, desde esta pantalla se puede acceder al conjunto de aplicaciones que componen RAVEN:

- El icono del “CALENDARIO” abrirá la aplicación para los eventos.
- El icono del “CRONOMETRO” abrirá la aplicación para el temporizador.
- El icono del “MANDO XBOX 360” abrirá el juego Simon.

Además del acceso a estas funcionalidades se puede acceder a las funcionalidades de cuenta desde el menú:

- Se puede cerrar sesión
- Ver información de la cuenta
- Borrar cuenta del usuario.

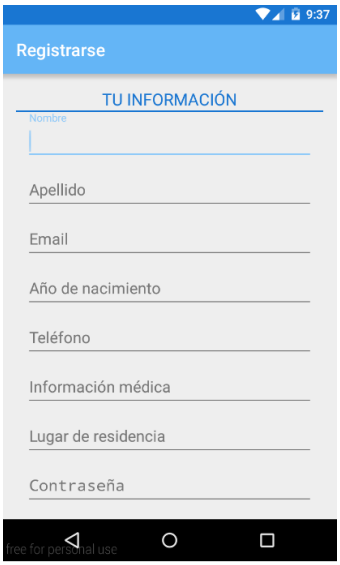


Figura 8: Pantalla de registro

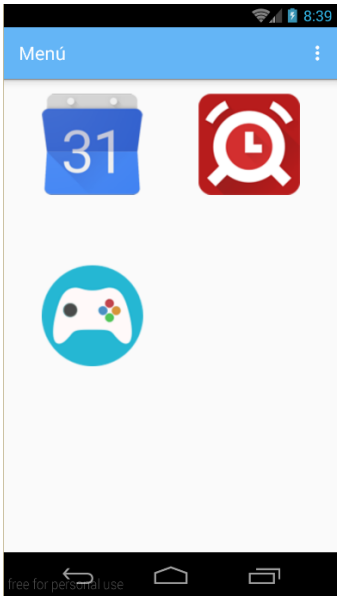


Figura 9: Pantalla menú

A.2.4. Pantalla eventos/calendario

Esta es la pantalla de los eventos, a la que se accede desde el icono “CALENDARIO” de la pantalla menú, en esta pantalla se muestra un “CALENDARIO NAVEGABLE”, una “LISTA DE EVENTOS” y un botón “CREAR EVENTO”.

En el “CALENDARIO NAVEGABLE” se puede desplazar por meses arrastrando el dedo, pulsar en día cualquiera y muestra días con eventos asignados mediante un círculo en el día.

En la “LISTA DE EVENTOS” se muestran los eventos del día seleccionado en el “CALENDARIO NAVEGABLE”, esta lista se actualizará conforme se cambie el día seleccionado. En esta lista se puede pulsar sobre cualquier “EVENTO” se abrirá una pantalla donde se puede visualizar los datos del mismo e incluso modificarlos. Si en vez de pulsarlo, se mantiene pulsado el “EVENTO” nos da opción de borrarlo.

El botón “CREAR EVENTO” nos abrirá una pantalla que se mostrará a continuación, pantalla evento.

Figura 10: Pantalla eventos/calendario

A.2.5. Pantalla evento

Esta pantalla es la encargada de la información de los eventos, se utiliza tanto para la creación como la modificación de los “EVENTO”.

En la modificación esta pantalla tendrá un evento asignado con los datos del evento ya rellenados, los cuales se pueden modificar sin ningún problema.

En la creación la pantalla sale vacía de datos y deberemos rellenarla, es obligatorio el mensaje. Tenemos 2 tipos de “EVENTO”:

- PUNTUAL en la que se debe especificar el mensaje, fecha y hora del “EVENTO” y no se debe de marcar ninguno de los días de los campos de selección → L M X J

V S D. Una vez creado se mostrará en la pantalla eventos/calendario en el día que se haya especificado.

- PERIÓDICO en la que se debe especificar el mensaje, los días que deseemos de los campos de selección → L M X J V S D. Una vez creado se mostrará todas las semanas los días seleccionados.

Figura 11: Pantalla evento

A.2.6. Pantalla contador

Esta es la pantalla de los eventos, a la que se accede desde el icono "CONTADOR" de la pantalla menú, en esta pantalla se muestra un campo de texto "TITULO", "TIME SCROLL" y un botón "CREAR CONTADOR".

Para crear un contadores obligatorio rellenar el campo "TITULO" que será la información que nos facilitara la notificación, y el "TIME SCROLL" con tiempo mayor a 0 segundos. El "TIME SCROLL" cuenta con tres columnas, de derecha a izquierda son: segundos, minutos, horas. Una vez introducidos los campos se puede pulsar el botón "CREAR CONTADOR". Una vez creado el contador estará ya activo, cuando se haya cumplido el tiempo especificado nos saltará una notificación con el título aportado.

A.2.7. Pantalla simon

Esta es la pantalla del juego simon, a la que se accede desde el icono "JUEG' de la pantalla menú, en esta pantalla se muestra muestra una circunferencia compuesta por 4 botones o cuadrante de colores y en la parte baja un botón de inicio / start.

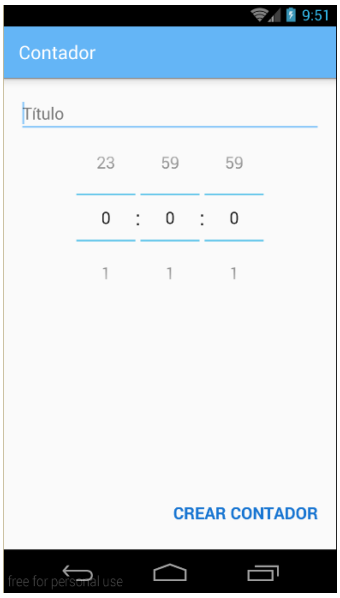


Figura 12: Pantalla contador

Para comenzar a jugar se pulsará al botón start, e inmediatamente comienza el juego.

Como jugar, el juego de forma aleatoria va iluminando los cuadrantes de colores, y a la vez que se ilumina cada cuadrante emite un sonido propio. Después de esperar, el usuario debe ir introduciendo la secuencia mostrada en el orden correcto, ayudándose de su memoria visual y sonora. Si lo consigue, éste responderá con una secuencia más larga, y así sucesivamente. Si falla, el usuario debe volver a empezar.

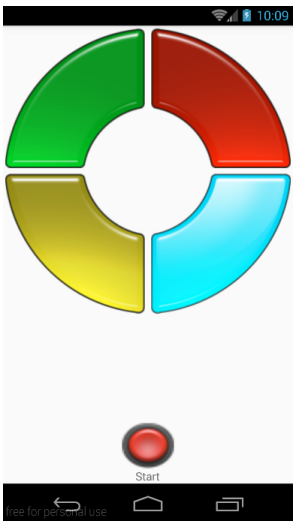


Figura 13: Pantalla simon