

Informe Primer Sprint

HAY QUE PONER INTERFACES CHAVALES

Raven

"Ayudarte a recordar"



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

Gestión de Proyecto Software
Grado en Ingeniería Informática

Índice

| | |
|---|-----------|
| 1. Introducción..... | 3 |
| 1.1. Descripción del proyecto..... | 3 |
| 1.2. Descripción del equipo..... | 3 |
| 2. Producto..... | 4 |
| 2.1 Plan de producto..... | 4 |
| 2.2. Planificación de lanzamientos..... | 4 |
| 2.3. Análisis de riesgos..... | 5 |
| 2.4. Definición de hecho..... | 5 |
| 2.5. Pila del producto en el estado actual..... | 6 |
| 2.6. Historias de usuario..... | 7 |
| 2.7. Criterios de aceptación..... | 8 |
| 2.8. Bocetos de la GUI..... | 11 |
| 2.9. Estado actual de la aplicación..... | 11 |
| 2.10. Arquitectura y diseño..... | 13 |
| 3. Proceso..... | 16 |
| 3.1. Tests..... | 16 |
| 3.2. Control de versiones..... | 17 |
| 3.3. Diagramas de trabajo completado..... | 18 |
| 3.4. Esfuerzos por actividades..... | 18 |
| 3.5. Esfuerzos por personas..... | 19 |
| 3.6. Otros aspectos de la gestión del proyecto..... | 21 |
| 3.7. Medidas para mejorar el proyecto..... | 21 |
| 4. Conclusiones..... | 22 |

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 al 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 distintos ámbitos: bases de datos, desarrollo de servidores, aplicaciones Android, administración de sistemas y control de versiones.

Los miembros del equipo son:

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

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 podrán 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 | <ul style="list-style-type: none">• Creación de eventos en el calendario• Marcación rápida de llamadas.• Servidor• Cronómetro y temporizador.• Cuentas de usuario.• Juego <i>simón</i>. | 1 año |
| 2º | <ul style="list-style-type: none">• GPS• Juego <i>sudoku</i> | 6 meses |
| 3º | <ul style="list-style-type: none">• Juego <i>memory</i>• 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 | Contingencia |
|---|--------------|-------|---|
| Perdida de información. | Baja | Alto | Realización de copias de seguridad periódicas |
| 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 que 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 un maquina 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 considerará que un hecho está completado cuando:

- Esta 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

Estado actual de la pila de producto

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

| Prioridad | PBI | Tamaño | Completado |
|-----------|--------------------------------|--------|------------|
| 1 | Crear usuario cuenta | MM | Sí |
| 2 | Log in/out | SS | Sí |
| 3 | Crear calendario | M | Sí |
| 4 | Crear evento calendario | S | Sí |
| 5 | Visualizar eventos calendarios | M | Sí |
| 6 | Visualizar evento | M | Sí |
| 7 | Visualizar usuario cuenta | S | Sí |
| 8 | Juego <i>simon</i> | LL | No |
| 9 | Temporizador | L | No |
| 10 | Borrar evento calendario | M | No |
| 11 | Borrar usuario cuenta | S | No |
| 12 | Modificar evento calendario | M | No |
| 13 | Modificar usuario cuenta | S | No |
| 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 <i>memory</i> | [] | No |
| 21 | Juego <i>sudoku</i> | [] | No |
| 21 | Localización GPS | XL | No |

En este momento no se ha realizado el *grooming* para la realización del segundo sprint del proyecto, por lo que puede que se realice cambios sobre la pila.

2.6. Historias 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.

2.7. Criterios de aceptación

Los criterios de aceptación para los PBI desarrollados durante el primer spring 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.
- Tras la creación de una cuenta de usuario está se vera reflejada en la base de datos.
- El servidor debe contar con una interfaz para la creación del usuario.
- Existe test automatizados que permite comprobar que el servidor responde correctamente a las peticiones de crear un 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.
- El servidor cuenta con una interfaz para la creación del usuario.
- Existe test automatizados que permite comprobar que el servidor responde correctamente a las peticiones de crear un usuario.

Crear calendario:


- Se dispone una GUI, en la aplicación Android, que muestra un calendario. 

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.
- El servidor contará con una interfaz, a la cual se enviará los datos de un evento desde la aplicación.
- Tras la creación del evento esté quedará reflejado en la base de datos.
- Existen test automatizados que comprueban que el servidor funciona correctamente para la creación del evento.

Visualizar evento calendario:

- Los días en los cuales hay un evento están reflejados en el servidor 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.
- Se dispone de una interfaz en el servidor que permite pedir el listado de eventos de un día para un usuario concreto. 

Visualizar evento:

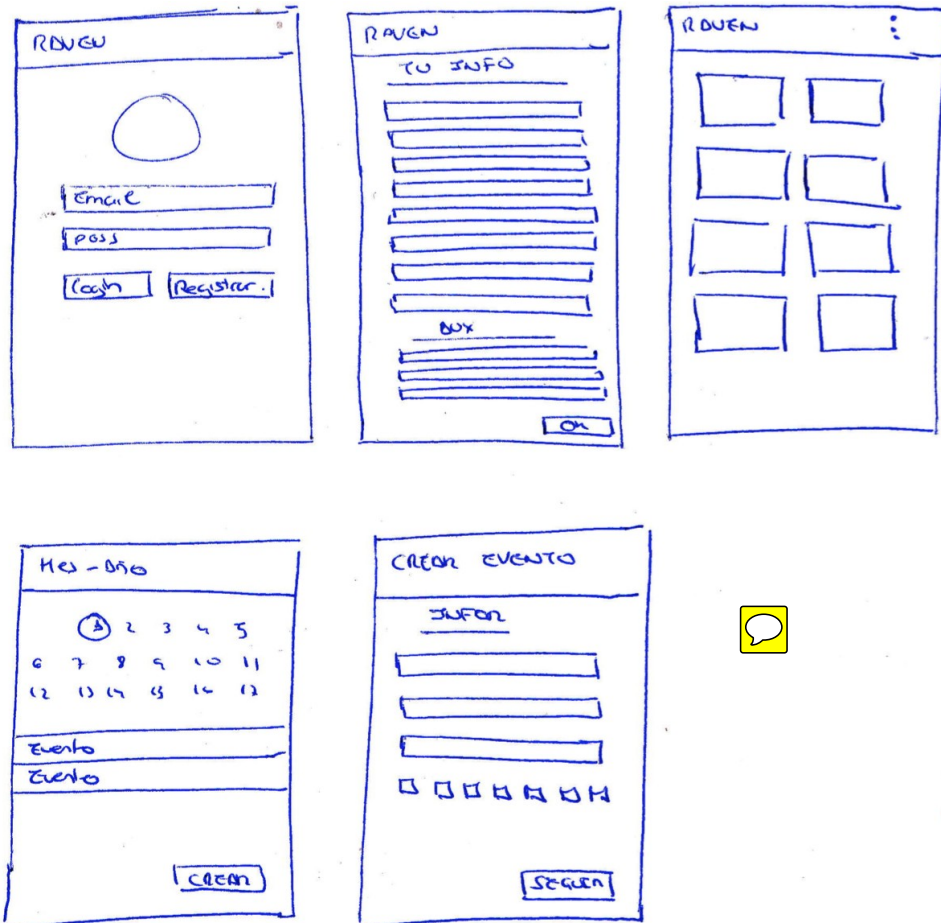
- Se dispone de una GUI en la cual se puede visionar con detalle la información de un evento: el mensaje, fecha, hora y días en los que se repite.
- Existe una interfaz en el servidor que permite pedir un evento concreto mediante su id.
- Se puede acceder a la visualización de un evento concreto, a la 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 -.
- Existe una interfaz en el servidor que permite pedir la información del usuario.
- 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.

2.8. Bocetos de la GUI

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



2.9. Estado actual de la aplicación

Funcionalidad implementada

Actualmente se ha implementado la funcionalidad más básica de la aplicación:

- Creación de usuario.
- Log in/log out del usuario.
- Visualización de la información del usuario.

- Creación del calendario.
- Creación de eventos.
- Visualización de eventos en el calendario.
- Visualización de eventos en detalle.

GUI implementada hasta el momento

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

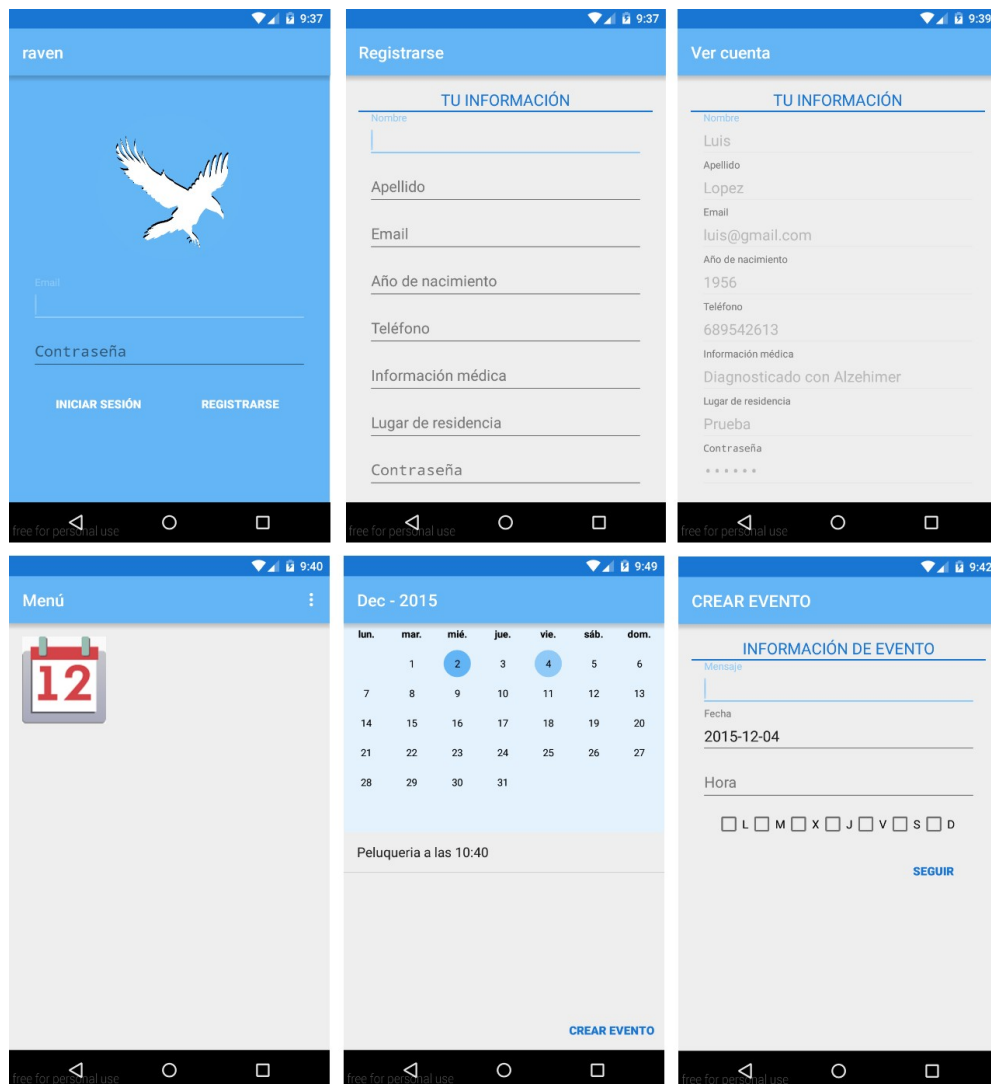


Figura (de izquierda superior a derecha inferior): 1. Pantalla de Login, 2. pantalla de registro, 3. visualización de información, 4. pantalla principal de la aplicación, 5. calendario de la aplicación con visualización de eventos, 6. pantalla de creación/visualización en detalles de eventos.

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

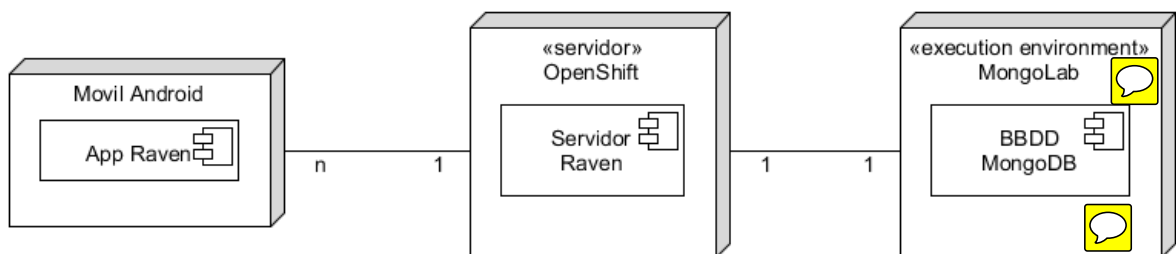
Problemas conocidos

Tras la realización del primer sprint se encontraron ciertos problemas en la aplicación que, si está de acuerdo el dueño de producto, se podrán solucionar en el segundo sprint:

- Los mensajes mostrados al usuario en caso de error deberían ser más informativos.
- El color de la aplicación quizás no es el más adecuado para el tipo de cliente al que va dirigido.
- Habría que establecer un botón extra para tener la capacidad de volver atrás.

2.10. Arquitectura y diseño

Despliegue

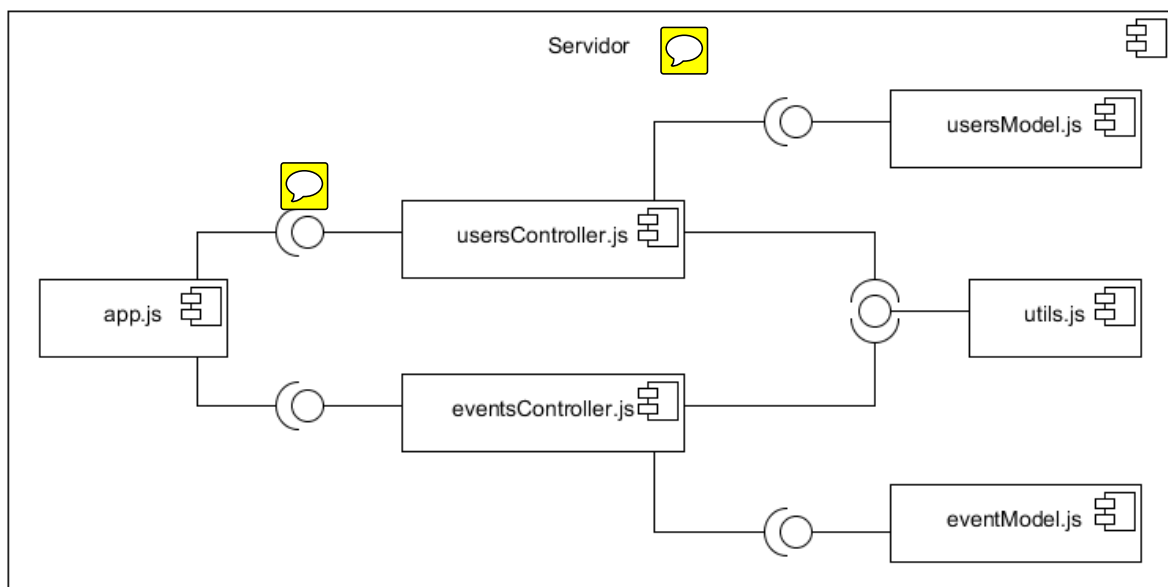


Se ha optado por el siguiente diagrama de despliegue debido a que presenta los siguientes puntos fuertes:

- Se trata de una aplicación que sigue una estructura de cliente-servidor.
- Se ha separado el servidor y la base de datos en dos máquinas diferentes con el fin de conseguir una mayor escalabilidad.
- El tener un único servidor permite la centralización de los recursos, y que en una primera instancia, consiguiendo que la integración sea sencilla.

- Facilidad de mantenimiento e integración de todos los servicios.
- Puede escalar fácilmente al poder aumentar el número de servidores.
- Es fácil conseguir tolerancia a fallos. 💬
- Es un estructura conocida por todos los desarrolladores de la aplicación.

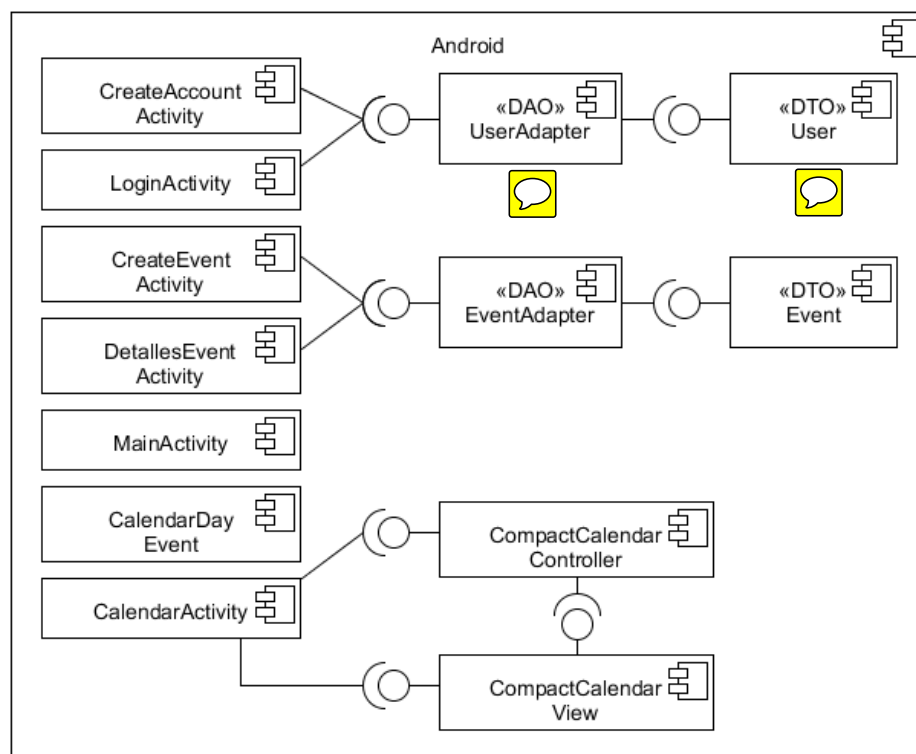
Componentes del servidor



- 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 muchos 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 desarrollo, y que nuevos desarrolladores se puedan unir posteriormente al proyecto sin grandes complicaciones.

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.

3 Proceso

3.1. Tests

Servidor

Se ha decidido 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.

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

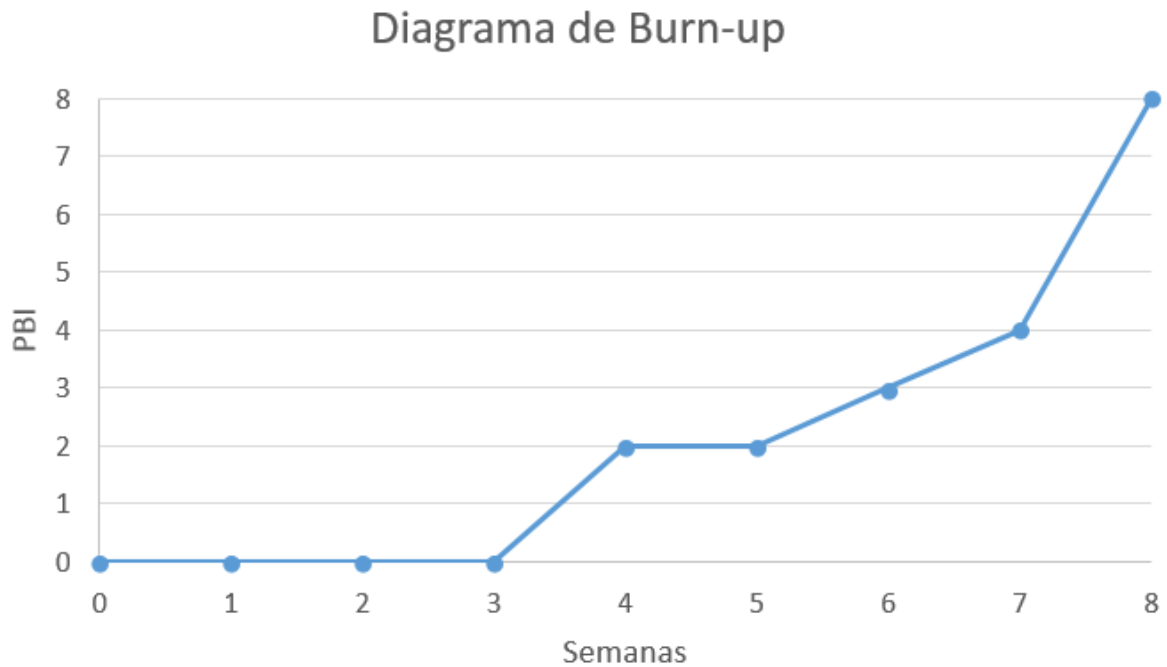
Para el control de versiones se ha utilizado Git y GitHub, en el cual se ha optado por la creación de una organización con dos repositorios:

- 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 decidido 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.



3.4. Esfuerzos por actividades

| 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. |
| | | 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. |

3.5. Esfuerzos por personas

El esfuerzo realizado 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

| 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 | 2 |
| | 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 de la aplicación. | 0 | 2,5 | 0 | 0 |
| | Lógica del servidor. | 0 | 0 | 2 | 1 |
| | Test automatizados. | 0 | 2 | 1 | 1 |
| | Protocolo de comunicación. | 0 | 1 | 1 | 1 |
| | Documentación. | 0 | 0 | 0 | 1 |

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 proyecto

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 relevantes del primer sprint. Destacando de ellos:

1. Se ha logrado realizar todos los PBI planificados, pasando estos los test de aceptación y definición de hecho, en un tiempo un poco superior al esperado.
2. Se ha realizado un análisis del producto ha desarrollar analizando y planificando en profundidad los PBI ha realizar en el sprint correspondiente y dejando más difusos aquellos que se realizaran posteriormente.

Se ha buscado durante el desarrollo del proyecto llegar a la notificación de sobresaliente en los cuales se ha llegado a los siguientes objetivos:

Sobresaliente:

1. El código del proyecto se aloja en GitHub. Ver el repositorio de la aplicación (<https://github.com/UNIZAR-30248-2015-Raven>)
2. La cobertura de test alcanza el 30% la cobertura se presentará en persona, y esta medida mediante herramientas de cobertura como istanbul o codecov.
3. Definición de hecho clara, ver apartado Producto - Definición de hecho del documento.
4. Documentación arquitectural adecuada y justificada, ver apartado Producto - Arquitectura y diseño del documento.
5. Cumplimento suficiente de los requisitos.
6. Se ha llevado una hoja de esfuerzo entregada por persona.
7. La compilación y gestión de dependencias esta basada en scripts: observar los scripts de granel para la aplicación android y npm para el servidor.

8. Toda la documentación del proyecto esta bajo control de versiones: la documentación se encuentra en la wiki del proyecto (<https://github.com/UNIZAR-30248-2015-Raven/Servidor/wiki>) y el control de versiones de dropbox.
9. Todas las historias de usuario tienen criterios de aceptación, ver apartado Producto - Criterios de aceptación del documento
10. Hay test de aceptación automáticos para al menos 1 criterio de aceptación, login y creación de evento, puede verse la realización del test en el código y se realizara una prueba en persona.
11. La construcción se ha automatizado, existe un trigger automático mediante travis para pasar los test al realizar un push, además de lanzamiento automático de test, etc.