UOC Universitat Oberta de Catalunya

Stori

Aplicación móvil para la creación de historias de forma colaborativa.

Autor: **Daniel Franco Cecilia**

Consultor: Jordi Ceballos Villach

A todas las personas que me han
apoyado y se han preocupado por mí durante este tiempo,
especialmente a mi familia.

Resumen

El gran auge de las tecnologías móviles en los últimos años ha supuesto un cambio radical en la creación de aplicaciones para millones de desarrolladores. La aparición de los smartphones y las tablets han traído consigo la creación de nuevos mercados a disposición de los programadores de todo el mundo. Las grandes compañías tecnológicas como Apple, Google, Nokia, Microsoft etc... han sido capaces de tanto directa como indirectamente hacerse un hueco en este nuevo mercado creando software y plataformas que permiten la interacción entre desarrolladores y usuarios.

Apple ha conseguido crear un ecosistema de software (iOS) y hardware (Iphone, Ipad) que permite la intercomunicación de sus dispositivos de una forma muy fácil y agradable al usuario. Los usuarios de Apple pueden comunicar sus dispositivos, acceder a servicios de computación en la nube, descargar aplicaciones etc... y todo ello ofrecido por la misma empresa.

Google es el responsable del desarrollo de Android, el otro sistema operativo más relevante en los smartphones actuales. Google no dispone hasta el momento de un ecosistema de software y hardware tan completo como el de Apple. Múltiples fabricantes de hardware instalan Android en sus dispositivos dado que no es un software cerrado como el de Apple.

Nokia fabrica el hardware e instala en sus dispositivos el sistema operativo de Microsoft Windows Phone. Es un caso distinto a los dos anteriores ya que en este caso existe un acuerdo entre dos grandes empresas en el que uno fabrica el hardware y otro el software.

Estos son los actores más importantes cuando se habla de la creación de software para dispositivos móviles actualmente. Pero hay otras empresas que ya han anunciado nuevos sistemas operativos (Mozilla, Canonical) y por lo tanto el futuro se presenta cuanto menos interesante.

Tras la presentación del contexto actual que corresponde al área en el que se desarrolla este trabajo final de carrera, hay que resumir los puntos clave de los que se compone éste:

- *Servidor node.js:* El servidor implementa toda la lógica de comunicación de usuarios en tiempo real, gestiona la conexión y desconexión de éstos, el almacenamiento temporal en base de datos de los usuarios conectados (MongoDB), la gestión de las "rooms" de creación de historias, la gestión de los turnos de los usuarios conectados etc...
- *Cliente móvil (app):* La app está desarrollada en HTML5+CSS+JavaScript. Se ha utilizado Jquery Mobile como framework de desarrollo JavaScript y Apache Cordova como framework de desarrollo para el acceso a las funcionalidades de los dispositivos móviles.

La conjunción de estos dos componentes principales forma el sistema de creación de historias de forma colaborativa, que permite al usuario crear historias de diferentes tipos y géneros con otros usuarios de la aplicación

Índice de contenido

1. Introducción

1.1. Justificación y contexto del proyecto

El presente trabajo final de carrera se realiza en el contexto de las aplicaciones de dispositivos móviles. De los distintos escenarios de aplicación propuestos para la realización del TFC se ha optado por el desarrollo de una aplicación móvil multiplataforma para la creación de historias de forma colaborativa usando las tecnologías que nos brinda HTML5.

La elección de la tecnología se ha basado en el conocimiento previo en éstas tecnologías y el interés en el aprendizaje más profundo debido al gran futuro que se les augura. Además se ha incorporado un componente más que añade valor al producto final desarrollado, el servidor. El servidor es una parte fundamental y que contiene una gran carga de la lógica de negocio de este proyecto. La tecnología a utilizar en la creación del servidor es *node.js.* Cómo en el caso anterior se ha decidido por criterios de afinidad y de interés y sopesando el trabajo de investigación previo a realizar para conocer las herramientas a utilizar para obtener el mejor resultado.

La idea de la app surge de la intención de desarrollar una aplicación de entretenimiento, social y que implique un aprendizaje o un desarrollo de las capacidades del usuario. Se justifica pues, ya que la creación de pequeñas historias conlleva una práctica del lenguaje escrito y la estimulación de la creatividad. Es una aplicación que puede ser utilizada por cualquier usuario de cualquier edad, aunque puede resultar especialmente útil como herramienta de aprendizaje para los niños.

1.2. Descripción del proyecto

El presente proyecto tiene por objeto realizar el análisis, diseño e implementación de una aplicación multiplataforma para dispositivos móviles que se conecta a un servidor para la gestión de la lógica de la creación de historias con otros usuarios, utilizando node.js para el desarrollo del servidor y HTML5+CSS3+Javascript y Apache Cordova para el desarrollo de la aplicación cliente móvil.

La comunicación entre las aplicaciones cliente y el servidor se realizará mediante el protocolo de comunicación *Websocket.* Esta tecnología proporciona un canal de comunicación bidireccional sobre un único socket TCP. Se adapta especialmente bien al proyecto ya que el servidor debe manejar muchos clientes conectados que a su vez envían información a otros clientes conectados, todo esto en tiempo real. La programación en servidor se apoya en el módulo socket.io que proporciona una *API* para el desarrollo con Websocket y que cuenta además con una gestión de *rooms* de usuarios muy útil para este proyecto. Para las operaciones contra la base de datos MongoDB del servidor se utiliza el módulo mongojs.

A continuación se proporciona una explicación de cada una de estas tecnologías con el fin de presentarlas en mayor detalle:

WebSocket: Este protocolo de comunicación proporciona una comunicación full-duplex sobre un único socket TCP. Es decir, es capaz de enviar y recibir mensajes de forma simultánea, lo cual es perfecto para aplicaciones como *Stori,* que se basa en la comunicación en tiempo real entre diferentes usuarios. Actualmente la API de WebSocket está siendo estandarizada por el W3C y el protocolo WebSocket por el IETF.

W3C (World Wide Web Consortium) es un consorcio internacional que desarrolla especificaciones técnicas y directrices para todas las actuaciones técnicas sobre la red. El IETF (Internet Engineering Task Force) es una organización internacional abierta de normalización que regula y controla las propuestas y estándares de Internet. Según indican en su web el objetivo de esta organización es que Internet funcione mejor produciendo documentación técnica y relevante de alta calidad que influya en la forma en que las personas diseñan, usan y gestionan Internet.

Socket.io es la librería JavaScript utilizada en este proyecto para programar toda la gestión entre los clientes conectados en tiempo real. Socket.io utiliza el protocolo WebSocket y facilita la programación sobre este protocolo además de añadir funcionalidades como la posibilidad de enviar mensajes a todos los clientes conectados, el almacenar información sobre los clientes conectados, el agrupar los clientes en rooms y una entrada/salida asíncrona.

MongoDB es una base de datos NoSQL que en el proyecto se utiliza en el servidor para almacenar temporalmente los usuarios que se conectan. Las características principales de esta base de datos son las siguientes:

- *Almacenado orientado a fichero/documento*: Documentos de tipo JSON con esquemas dinámicos que ofrecen simplicidad y mucha capacidad de adaptación a cambios.
- *Soporte Full index*: Índice en cada atributo
- *Replicación*: Soporta el tipo de replicación maestro-esclavo. El maestro puede leer y escribir y el esclavo puede copiar los datos del maestro pero sólo se puede utilizar para lectura o copias de seguridad.
- *Balanceo de carga:* Se puede escalar de forma horizontal mediante la utilización del concepto shard que permite elegir una llave shard que determine cómo serán distribuidos los datos en una colección. MongoDB puede ejecutarse en varios servidores, balanceando la carga y/o duplicando los datos para poder mantener el sistema funcionando en cualquier caso.
- *JavaScript del lado del servidor:* Permite la ejecución de consultas mediante JavaScript.

Los usuarios introducirán un email y una contraseña que los identificará como usuarios tanto en el servidor como en la app. Esta información será almacenada en una pequeña base de datos local. Así pues, sólo se pide esta información en el primer uso de la aplicación. En usos posteriores no hará falta, ya que se accederá a la información almacenada en la base de datos.

Una vez el usuario ejecuta la aplicación estará en disposición de crear una historia con los contactos que estén utilizando la aplicación en ese momento o de consultar las historias creadas con anterioridad. Cuando un usuario crea una historia, éste la podrá guardar o descartarla. Las historias se almacenan en la base de datos local para su posterior consulta en la app.

El diseño de la app se ha pensado con el fin de ser simple, con un interfaz gráfico agradable y que cumpla perfectamente la función con la que se ha creado, que es la creación de historias con otros usuarios. Para ello se ha utilizado Jquery Mobile y se ha diseñado un *theme* específico.

1.3 Objetivo

El objetivo de este proyecto es el desarrollo de una app para dispositivos móviles con el fin de crear historias de forma colaborativa. El proceso de creación de la historia será por turnos, permitiendo a cada participante realizar su aportación o pasar turno. Además se podrán almacenar las historias en las que el usuario ha participado, siendo posible luego consultarlas o continuarlas en caso de no estar finalizadas.

La finalidad principal de la app es el entretenimiento. Pretende ser una pequeña plataforma que sirva para dar alas a la creatividad tanto de personas aficionadas a la escritura como para aquellos que utilizan sus dispositivos móviles para jugar con otros.

La aplicación pretende además tener una función educativa. La creación de historias implica la práctica de la escritura y estimula la creatividad y la imaginación. Así pues, a nivel educativo puede resultar de gran ayuda como actividad extra tanto para niños como para adultos que deseen mejorar esos aspectos al mismo tiempo que se divierten.

1.4. Listado de funcionalidades

A continuación se detallan el listado de funcionalidades desarrolladas agrupadas por subsistema:

Subsistema de gestión de historias

- Listado de "Mis historias"
- Consulta de la ficha de cada historia
- Almacenamiento de historias en la base de datos local
- Editar historia
- Eliminar historias
- Crear nueva historia
- Selección múltiple de contactos

Subsistema de comunicaciones

- Gestión de invitaciones entre usuarios para crear historias
- Gestión de las "salas" de creación de historias
- Gestión de turnos
- Gestión de notificaciones a los usuarios
- Comunicación clientes-servidor

1.5. Tecnología a utilizar

Se ha decidido utilizar las siguientes tecnologías para desarrollar el producto:

App móvil

Para acceder a la funciones del dispositivo móvil se hará uso del framework de desarrollo Apache Cordova. El lenguaje a utilizar será HTML5 + CSS3 + JavaScript. Para el desarrollo del frontal se hará uso del framework de desarrollo Jquery Mobile.

Los motivos de la elección de esta tecnología la parte de desarrollo móvil son:

- *Multiplataforma:* Esta tecnología permite desarrollar una vez y desplegar el producto en muchos dispositivos móviles distintos.
- *Disminución de los problemas de fragmentación:* La fragmentación es uno de los mayores problemas para los desarrolladores. La cantidad de resoluciones diferentes hace muy complicado el desarrollo de las pantallas de las aplicaciones. El utilizar HTML5 + CSS3 permite dar una solución elegante a este problema
- *Experiencia del desarrollador:* La experiencia del desarrollador en JavaScript y el especial interés por este lenguaje es una de las razones importantes por las que se ha decidido el uso de esta tecnología.

Servidor

Una parte fundamental de este proyecto es el desarrollo del servidor que permita establecer las comunicaciones entre los clientes de una forma satisfactoria. La tecnología a utilizar para desarrollar el servidor es *node.js*.

Los motivos de la elección de esta tecnología para la parte del desarrollo del servidor son:

- *Lenguaje:* Como se ha comentado anteriormente, el desarrollador tiene especial interés en todas las tecnologías nuevas que giran alrededor de JavaScript y node.js es una de ellas.
- *Rapidez:* Numerosos benchmarks muestran que node.js es muy rápido (event-loop, non-blocking)
- *Comunidad de desarrolladores activa:* Es una nueva tecnología que está creciendo muy rápidamente y está recibiendo mucho apoyo tanto de sponsors como de desarrolladores.
- *Arquitectura orientada a eventos (event driven architecture):* Esta arquitectura permite hacer aplicaciones más escalables
- *Rendimiento:* La propia filosofía de node.js hace que sea muy rápido y maneje muy bien múltiples conexiones.

1.6. Planificación

1.6.1 Ciclo de vida

El proyecto se ha desarrollado tomando como base el ciclo de vida clásico de creación de software (también denominado ciclo en cascada) adaptado a las necesidades del proyecto.

A lo largo de las distintas fases se ha ido generando documentación que ha sido entregada en los hitos establecidos inicialmente en el plan de trabajo. El contenido de éstos ha sido condensado en esta memoria final del trabajo final de carrera.

1.6.2 Detalle de actividades

Las siguientes secciones describen las principales actividades realizadas en cada etapa con el objeto de obtener un producto que cumpliese los objetivos requeridos de la mejor forma posible y con la mayor calidad.

1.6.2.1 Análisis previo de planificación

En esta etapa se ha hecho hincapié en obtener una visión global del sistema que queremos construir y de las actividades que lo van a componer

Actividad	Descripción
Selección del proyecto	Selección de proyecto propio
Preparación del proyecto	Lectura de documentación, preparación de entorno
Planificación del proyecto	Temporalización del proyecto según el calendario de hitos a entregar

Creación del documento	Generación del plan de trabajo
Envío PEC1	Envío del documento de esta etapa

1.6.2.2 Análisis de requisitos

Durante esta fase se han detallado las necesidades a cubrir por el software. Se han analizado los requisitos funcionales y no funcionales. También se ha desarrollado un pequeño prototipo de la parte de servidor, con unos clientes web de prueba para testear su funcionamiento.

Actividad	Descripción
Especificación de requisitos funcionales	Selección de proyecto propio
Especificación de requisitos no funcionales	Lectura de documentación, preparación de entorno
Creación de prototipo de servidor y clientes de testeo	Temporalización del proyecto según el calendario de hitos a entregar

1.6.2.3 Diseño

Durante esta etapa se ha ido definiendo como debía ser el diseño arquitectónico del sistema, cómo debía ser la implementación de cada uno de los componentes y las tecnologías a utilizará en cada caso.

Actividad	Descripción
Concrección del diseño arquitectónico del servidor	Arquitectura del servidor
Diseño arquitectónico del cliente. App móvil.	Diferentes partes (pantallas), comunicación entre ellas, transiciones etc..
Diseño de la persistencia en servidor	Entidades y lógica de la información a guardar en la base de datos de servidor
Diseño de la persistencia en cliente	Entidades y lógica de la información a guardar en la base de datos de cliente
Diseño de las interfaces de usuario	Aspecto de las pantallas de la aplicación cliente
Estudio de alternativas tecnológicas	Estudio de distintas tecnologías (librerías, frameworks etc) para desarrollar los componentes
Preparación del documento a entregar	Recopilación de información y preparación del documento de análisis de requisitos y diseño del sistema
Entrega PEC2	Envío de la PEC2

1.6.2.4 Implementación y pruebas

Se ha llevado a cabo la implementación del software teniendo en cuenta las especificaciones detalladas en la fase correspondiente además de las pruebas pertinentes.

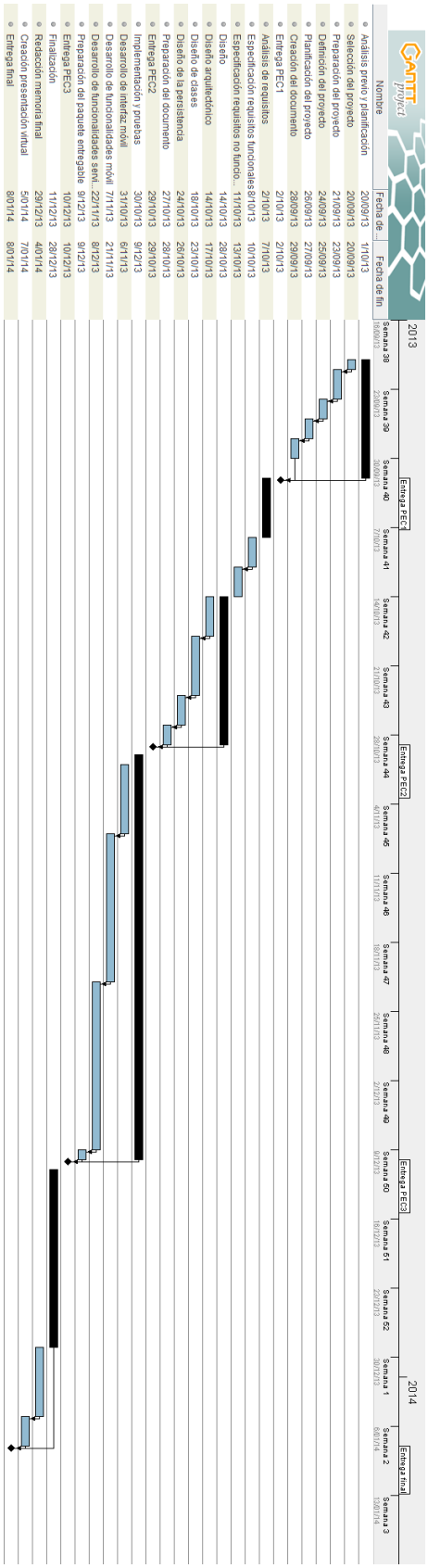
Actividad	Descripción
Desarrollo del servidor node.js	Implementación de las funcionalidades: • Acceso a base de datos • Entidad Usuario • Lógica de negocio • Gestión de “salas” • Gestión de turnos • Gestión de usuarios • Pruebas unitarias
Desarrollo de aplicación en dispositivos móviles	Implementación de las funcionalidades: • Diseño de interfaz gráfico • Componentes de conexión con el servidor node.js • Lógica de negocio • Pruebas unitarias
Preparación del paquete entregable	Creación del paquete de software
Envío de la PEC3	Entrega de la tercera PEC del trabajo

1.6.2.5 Finalización

Esta etapa no existe como tal. En su lugar se incluyen las fases de implementación, puesta en marcha, etc…En este caso, se ha utilizado para generar la documentación final del proyecto.

Actividad	Descripción
Redacción memoria final	Recopilación de documentos y redacción del documento final a entregar.
Creación presentación virtual	Creación de la presentación virtual
Entrega final	Entrega de la documentación

1.6.3 Calendario


GANTT project
Nombre
Fecha de ...
Fecha de fin
Análisis previo y planificación 20/09/13 1/10/13
Selección del proyecto 20/09/13 20/09/13
Preparación del proyecto 21/09/13 23/09/13
Definición del proyecto 24/09/13 25/09/13
Planificación del proyecto 26/09/13 27/09/13
Creación del documento 28/09/13 29/09/13
Entrega PEC1 2/10/13 2/10/13
Análisis de requisitos 2/10/13 7/10/13
Especificación requisitos funcionales 8/10/13 10/10/13
Especificación requisitos no funcio... 11/10/13 13/10/13
Diseño 14/10/13 28/10/13
Diseño arquitectónico 14/10/13 17/10/13
Diseño de clases 18/10/13 23/10/13
Diseño de la persistencia 24/10/13 26/10/13
Preparación del documento 27/10/13 28/10/13
Entrega PEC2 29/10/13 29/10/13
Implementación y pruebas 30/10/13 9/12/13
Desarrollo de interfaz móvil 31/10/13 6/11/13
Desarrollo de funcionalidades móvil 7/11/13 21/11/13
Desarrollo de funcionalidades servi... 22/11/13 8/12/13
Preparación del paquete entregable 9/12/13 9/12/13
Entrega PEC3 10/12/13 10/12/13
Finalización 11/12/13 28/12/13
Redacción memoria final 29/12/13 4/01/14
Creación presentación virtual 5/01/14 7/01/14
Entrega final 8/01/14 8/01/14
2013
2014
Semana 38
Semana 39
Semana 40
Entrega PEC1
Semana 41
Semana 42
Semana 43
Semana 44
Entrega PEC2
Semana 45
Semana 46
Semana 47
Semana 48
Semana 49
Semana 50
Entrega PEC3
Semana 51
Semana 52
Semana 1
Semana 2
Entrega final
Semana 3

1.7 Herramientas utilizadas

Para la realización del proyecto se han utilizado las siguientes herramientas:

Eclipse IDE: Entorno de desarrollo multiplataforma
Sublime Text: Editor de textos.
Firefox y Chrome: Para realizar las pruebas contra el servidor node.js.
LibreOffice: Para la redacción de documentos.
Gimp: Para la creación de gráficos

1.8 Productos obtenidos

La realización del trabajo fin de carrera ha generado los siguientes entregables:

- Documento del Plan de Trabajo
- Documento de Análisis y Diseño del sistema
- Servidor node.js para la intercomunicación de los clientes
- Aplicación móvil cliente
- Memoria final del proyecto
- Presentación virtual

1.9 Estructura del documento

El resto del documento recopila los aspectos más importantes de las tareas realizadas, empezando por los requisitos iniciales, que suponen el punto de partida del proyecto. Tras los requisitos se exponen los resultados de la fase de análisis. Posteriormente se detalla el diseño del sistema desde el punto de vista arquitectónico, las tecnologías utilizadas, diagramas que representan la arquitectura etc…

A continuación, se describe la fase de implementación, mostrando y explicando el código de algunas de las funcionalidades principales, y se explica el funcionamiento de la app. Y para finalizar se expondrán las conclusiones finales.

1.10 Riesgos del proyecto

Analizando las funcionalidades que se desean cumpla el proyecto final de carrera, podemos enumerar lo que se consideran los riesgos y dificultades que podemos esperar a lo largo del desarrollo.

Riesgo	Descripción	Probabilidad aparición	Impacto	Acciones mitigadoras
Tareas no programadas	El riesgo de tener que realizar más trabajo porque en la fase de planificación y análisis no se contempló el alcance total del proyecto.	Media	Alto	– Planear cuidadosamente las tareas – Establecer un plan de contingencia y de prioridades
Limitaciones tecnológicas	El encontrarse con límites que no permitan desarrollar lo que en un principio se diseño. Principalmente en el subsistema de comunicaciones.	Media	Alto	– Optar por alternativas – Cambiar la orientación final a conseguir con la app
Baja motivación	La motivación del desarrollador es imprescindible para obtener un buen resultado final. Una baja motivación afectaría negativamente al producto final.	Baja	Medio	– Centrarse en tareas pequeñas – Finalizar tareas con alta prioridad
Enfermedad	Problemas en la salud del desarrollador.	Baja	Medio	– Implicarse en otras tareas que no requieran de mucho esfuerzo intelectual
Desconocimiento del entorno y herramientas de desarrollo	La necesidad de investigar y formarse en las herramientas a utilizar en el desarrollo.	Alta	Crítico	– Preparar bien el entorno antes de empezar la fase de implementación
Fecha de entrega	El no llegar a la entrega final es el riesgo más importante. Todos los riesgos afectan éste.	Alta	Alto	– Hacer cambios que aunque reduzcan alguna funcionalidad permitan tener un producto bueno
Baja disponibilidad de terminales	Los dispositivos móviles para los que se desarrolla el proyecto son caros. El no disponer de varios terminales donde probar la aplicación final es un riesgo.	Alta	Crítico	– Realizar las pruebas con el equipo de desarrollo utilizando el navegador web

2. Análisis Funcional

En esta sección se van a analizar los requisitos funcionales y no funcionales del sistema que con la definición de los casos de uso nos darán una visión global del producto a realizar.

2.1 Requerimientos funcionales

La app móvil Stori tiene como objetivo la creación de historias entre usuarios por turnos. El sistema esta compuesto por la aplicación cliente móvil y el servidor node.js. A continuación se describirán las funcionalidades principales que harán posible esto.

2.1.1 Descripción básica del funcionamiento

Como ya se ha comentado, el propósito principal de esta app es la creación de historias por turnos en tiempo real. El funcionamiento básico se basa en los siguientes conceptos:

- *Elegir el tipo, género y los participantes de la historia a crear*: Es decir limitar la historia dotándola de características para clasificarla. El tener que elegir las características de la historia esta pensado también para posibles mejoras futuras de la aplicación. Una de las principales sería el hacer rankings por género, por tipo...etc.
- *Creación*: La pantalla de creación de la historia es donde ser produce la magia. Es donde se escribe la historia por turnos y donde se espera se de rienda suelta a la creatividad de los participantes.
- *Comunicación*: Esta app está pensada para ser social y por ello la creación de historias es en tiempo real, por turnos. La comunicación es un concepto clave en la aplicación.

2.1.2 Seguridad

Uno de los planteamientos principales al crear esta app era la de almacenar o no los usuarios en la base de datos del servidor. Al principio se pensó en que era lo mejor. El almacenar los usuarios y las historias que estos creaban en la base de datos del servidor permite ampliar las posibilidades de la aplicación pudiendo añadirle un componente social, compartir historias con todos los usuarios de la app, crear tops de historias, añadir ratings, añadir comentarios etc... mediante una web.

Pero en contra está el tema de la seguridad, ya que se debería almacenar información personal de muchos usuarios y protegerla de posibles ataques. Se decidió que lo mejor era centrarse en tener un producto final que cumpliera la función principal con la que se pensó, el escribir historias con tus contactos. Todo lo comentado anteriormente estará en la sección de futuras mejoras.

2.2 Requerimientos no funcionales

A continuación se tratarán algunos aspectos del sistema que no están relacionados con el proyecto como tal.

2.2.1 Interfaz de usuario

El interfaz de usuario de la app móvil se ha considerado un aspecto clave y como tal se ha tratado. En este tipo de aplicaciones es muy importante conseguir un diseño simple y usable a la par que agradable a la vista para el usuario.

Se le ha dado especial importancia al diseño de los iconos para hacer la app única. El escoger un color principal agradable y conseguir conjugar el resto de elementos de diseño con éste es una tarea difícil pero vital para obtener un resultado óptimo. Para ello se ha diseñado un theme para Jquery Mobile y se han creado iconos especialmente para la aplicación.

Por último hay que comentar que la gran fragmentación de dispositivos móviles en el mercado hace muy difícil el adaptar la interfaz a todo tipo de pantallas. Así pues no se puede garantizar la correcta adecuación de la interfaz gráfica a todos ellos. Durante la implementación se ha intentado minimizar este problema. Se garantiza el correcto funcionamiento en el terminal Samsung Galaxy SIII que ha sido el utilizado durante la etapa del desarrollo.

2.2.2 Uso de características propias de los dispositivos móviles

En el desarrollo de la app se ha hecho uso de algunas funciones propias de los dispositivos móviles a través del framework de desarrollo Apache Cordova. Concretamente se ha hecho uso del almacenamiento local en el dispositivo gracias a la base de datos local SQL Lite. Se ha utilizado para almacenar el usuario de la app y las historias que éste va creando con la app. Así pues hay dos tablas en la base de datos local. Una de usuario y otra de historias.

Otra de las funciones utilizadas en el desarrollo es el acceso a la lista de contactos del teléfono. Se accede al crear una nueva historia para listar los contactos y poder invitarlos a crear la historia. Hay que tener en cuenta que estos contactos deben tener el email con el que ese contacto se autentificó en la app.

2.3 Funcionalidades del sistema

A continuación se procede a describir las funcionalidades principales del sistema. Muchas de estas funcionalidades requieren de la comunicación entre clientes y servidor además del uso de algunas características propias de las plataformas móviles como se ha comentado en el apartado 2.2.2 Uso de características propias de los dispositivos móviles.

2.3.1 Cliente móvil

Autentificación de usuario: La primera vez que el usuario de la app acceda deberá autentificarse con un email y un password. Esta información se almacenará en una tabla de la base de datos local del Smartphone.

Listado de historias: Se trata de una pantalla que listará todas las historias que el usuario ha guardado en la base de datos local del smartphone. El usuario puede crear una nueva historia o ser invitado a participar en la creación de una historia iniciada por uno de sus contactos. En la pantalla donde se escribirá la historia habrá un botón para guardar la historia. En la pantalla de la ficha de la historia se podrá editar el contenido de una historia y guardarla.

Ficha de la historia: Será una pantalla para mostrar la información de la historia almacenada. Mostrará los datos de la historia (tipo, género…). En esta pantalla se podrá editar una historia que se hubiera creado con anterioridad o eliminarla de la base de datos local.

Crear nueva historia: Es una de las funcionalidades clave de la app. Esta funcionalidad permite crear una nueva historia. Para crear una nueva historia será necesario elegir el tipo, el género y los contactos con los que se desea crear la historia.

Tipo: Acotará las palabras de las que contará la historia a crear. Podrá ser corta, mediana o larga. La corta constará de 1000 palabras, la mediana será de 3000 y la larga de 5000.

Género: Se elegirá el género de la historia. La lista de géneros disponibles será:

- Terror
- Comedia
- Drama
- Thriller
- Misterio
- Romance
- Fantasía
- Ciencia ficción

Contactos: Se invitarán a los contactos con los que se quiere crear la historia. El número de contactos máximo a invitar dependerá del tipo de historia elegido anteriormente. Así pues si se elige de tipo corta, el máximo de participantes será dos, si se elige mediana el número de participantes será tres como máximo y si se elige larga el número de participantes será cuatro como máximo.

Escribir historia: Una vez creada la nueva historia, seleccionando el tipo, género e invitando a los contactos se procederá a invitar a los contactos seleccionados. Cuando los usuarios respondan a la

invitación aceptándola o rechazándola se podrá proceder a escribir la historia

La historia se escribirá por turnos, siendo el primer turno el del usuario creador de la historia. Un usuario puede escribir o pasar turno. Cuando el usuario envíe su aportación al servidor se pasará automáticamente de turno. Cada vez que un texto se envía, el número de palabras de la historia se va restando desde el máximo posible. Cuando la historia se termine, cada uno de los usuarios participantes podrá guardarla en su base de datos local.

2.3.2 Servidor

El servidor tiene que manejar las peticiones que se realizan desde la aplicación cliente. No dispone de una interfaz con la que el usuario interactúe. El funcionamiento es totalmente transparente para el usuario de la aplicación. Muchas de las funcionalidades de la aplicación cliente requieren llamar a una función del servidor.

A continuación se muestra una pequeña tabla con las operaciones en el cliente y el servidor:

Operación en cliente	Operación en servidor
Autentificación de usuario	Servidor. Autentificación de usuario
Enviar invitación	Servidor. Enviar invitación
Aceptar/Rechazar invitación	Servidor. Aceptar/Rechazar invitación
Escribir Historia	Servidor. Escribir historia

2.4 Usuarios del sistema

Nos podemos encontrar dos tipos de usuarios que se explicarán a continuación. Cabe decir que como se ha comentado con anterioridad se ha decidido reducir la gestión de usuarios a sólo la aplicación móvil con el fin de reducir amenazas de seguridad y centrar el desarrollo en el objetivo principal de la aplicación. Los usuarios del sistema son:

- *Usuario:* Es el usuario que inicia la aplicación por primera vez y por lo tanto tiene que registrarse introduciendo el email y la contraseña.
- *Usuario registrado:* Es el usuario que puede acceder a toda la funcionalidad de la app ya que está identificado con un email y una contraseña.

2.5 Casos de uso

Una vez identificados los actores que intervienen en la aplicación, podemos obtener una visión global de las funcionalidades de las que puede hacer uso cada uno de éstos.

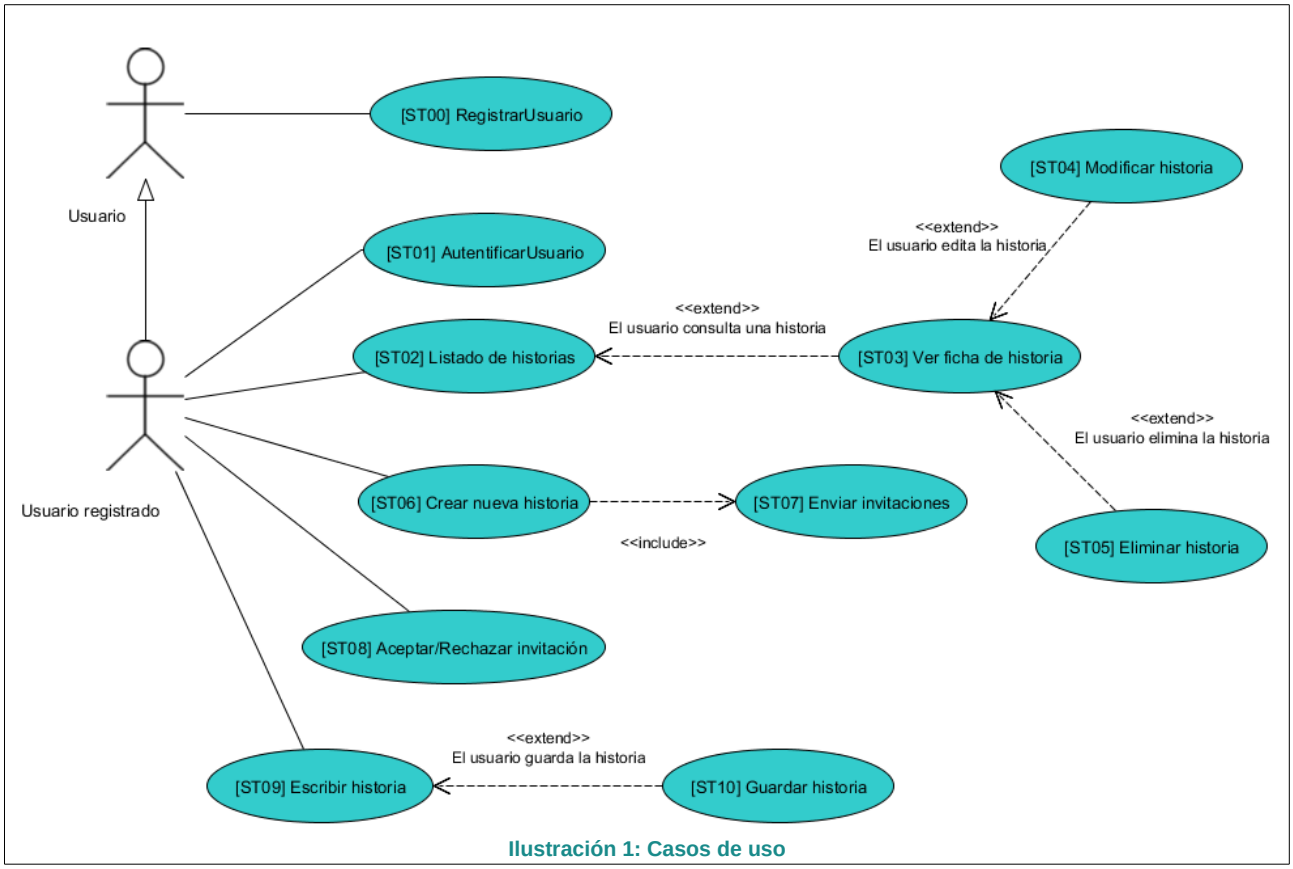


Ilustración 1: Casos de uso

En el diagrama anterior se pueden apreciar los dos actores existentes. El primero es el usuario la primera vez que entra en la aplicación. El segundo es el usuario una vez registrado. Es el actor principal ya que es el usuario capaz de acceder a todas la funcionalidades de la app.

Éstos son los casos de uso presentes en la app. Como se ha comentado con anterioridad, algunos de estos casos de uso llaman a funciones del servidor. Así pues a continuación se presenta un gráfico que ilustra la relación entre algunos casos de uso y las funciones del servidor que intervienen.

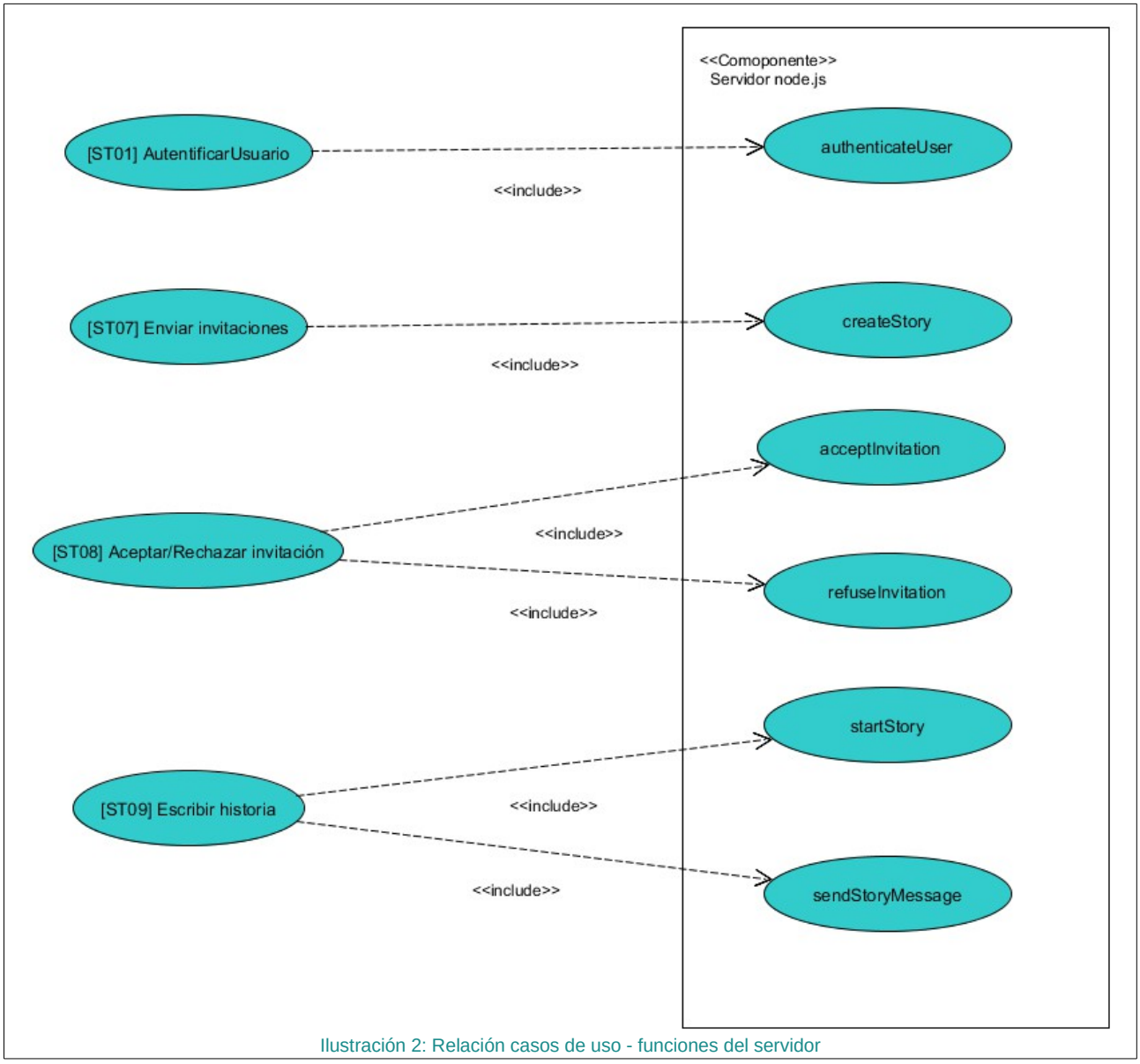


Ilustración 2: Relación casos de uso - funciones del servidor

Una vez presentados los casos de uso y las relaciones entre estos y las funcionalidades diseñadas en el servidor, se procede a describir individualmente cada uno de éstos de una forma más detallada con el fin de conocer las implicaciones y funcionamiento de cada uno de ellos:

2.5.1 Descripción de caso de uso. [ST00] Registro de usuario

Identificador:	ST00
Nombre:	Registro de usuario
Resumen:	La primera vez que el usuario entra en la aplicación tiene que registrarse dando su correo y contraseña.

Actores:	Usuario
Precondiciones:	El usuario no está registrado en el sistema
Postcondiciones:	El sistema almacena la información del usuario en la base de datos local.
Flujo normal:	1. El sistema chequea si existe el usuario en la base de datos 2. El usuario no existe. Se redirige a la pantalla de registro del usuario. 3. El usuario introduce la información en el formulario. 4. El sistema guarda la información en la base de datos local.
Flujos alternativos:	1. Se produce un error al guardar la información en la base de datos. Se muestra un error indicándolo
Inclusiones:	
Extensiones:	

2.5.2 Descripción de caso de uso. [ST01] Autentificar usuario

Identificador:	ST01
Nombre:	Autentificar usuario
Resumen:	Cuando un usuario registrado accede al sistema, éste se comunica con el servidor enviando la información del usuario para su almacenamiento en la base de datos del servidor mientras el usuario esté conectado.
Actores:	Usuario registrado
Precondiciones:	El usuario está registrado
Postcondiciones:	El usuario está almacenado temporalmente en la base de datos del servidor
Flujo normal:	1. El sistema envía la información del usuario al servidor 2. El servidor autentifica el usuario, almacenandolo temporalmente en la base de datos 3. El servidor devuelve un OK
Flujos alternativos:	1. Se produce un error al enviar o almacenar la información en la base de datos. Se muestra un mensaje de error.
Inclusiones:	
Extensiones:	

2.5.3 Descripción de caso de uso. [ST02] Listado de historias

Identificador:	ST02
Nombre:	Listado de historias
Resumen:	Se listarán las historias guardadas en la base de datos local
Actores:	Usuario registrado
Precondiciones:	El usuario está registrado
Postcondiciones:	El usuario recibe un listado de las historias alamacenadas
Flujo normal:	1. El usuario presiona el botón de listar historias

	2. El sistema consulta las historias almacenadas en la base de datos 3. El sistema muestra el listado de las historias
Flujos alternativos:	1. Se produce un error al obtener las historias. Se muestra un mensaje de error indicándolo.
Inclusiones:	
Extensiones:	**[ST03] Ver ficha de historia**

2.5.4 Descripción de caso de uso. [ST03] Ver ficha de la historia

Identificador:	ST03
Nombre:	Ver ficha de la historia
Resumen:	Se muestra la información de la historia seleccionada en el listado de historias
Actores:	Usuario registrado
Precondiciones:	El usuario está registrado El usuario ha seleccionado una historia del listado
Postcondiciones:	El usuario obtiene la información sobre la historia seleccionada
Flujo normal:	1. El usuario selecciona una historia del listado 2. El sistema consulta en la base de datos la historia seleccionada y devuelve la información de la historia 3. Se muestra la información recuperada en la pantalla de la ficha de la historia
Flujos alternativos:	1. Se produce un error al obtener la información de la base de datos. Se muestra un mensaje de error.
Inclusiones:	
Extensiones:	**[ST04] Modificar historia** **[ST05] Eliminar historia**

2.5.5 Descripción de caso de uso. [ST04] Modificar historia

Identificador:	ST04
Nombre:	Modificar historia
Resumen:	Se modifica el contenido de la historia y se guarda en la base de datos
Actores:	Usuario registrado
Precondiciones:	El usuario está registrado El usuario ha seleccionado una historia
Postcondiciones:	Historia modificada almacenada en la base de datos
Flujo normal:	1. El usuario modifica el contenido de la historia seleccionada 2. Pulsa el boton de guardar y la historia se almacena en la base de datos
Flujos alternativos:	1. Se produce un error al almacenar la historia en la base de datos. Se muestra un mensaje de error.
Inclusiones:	
Extensiones:	

2.5.6 Descripción de caso de uso. [ST05] Eliminar historia

Identificador:	ST05
Nombre:	Eliminar historia
Resumen:	Se elimina la historia de la base de datos local
Actores:	Usuario registrado
Precondiciones:	El usuario está registrado La historia existe en la base de datos
Postcondiciones:	La historia es eliminada de la base de datos
Flujo normal:	1. El usuario pulsa el botón de eliminar y la historia es borrada de la base de datos
Flujos alternativos:	1. Se produce un error al borrar la historia de la base de datos. Se muestra un mensaje de error.
Inclusiones:	
Extensiones:	

2.5.7 Descripción de caso de uso. [ST06] Crear nueva historia

Identificador:	ST06
Nombre:	Crear nueva historia
Resumen:	Se crea una historia del tipo y género seleccionados con los usuarios invitados
Actores:	Usuario registrado
Precondiciones:	Se selecciona un tipo, género y se invitan a usuarios a crear la historia
Postcondiciones:	Se escribe la historia
Flujo normal:	1. Se pulsa el botón de crear historia 2. Se selecciona un tipo 3. Se selecciona un género 4. Se seleccionan los contactos con los que se quiere crear la historia 5. Se envían las invitaciones 6. Se comienza a escribir la historia, si el número de usuarios es suficiente
Flujos alternativos:	1. Si el número de usuarios que han aceptado crear la historia no es suficiente entonces se rechaza la creación de la historia y se muestra un mensaje.
Inclusiones:	**[ST07] Enviar invitaciones**
Extensiones:	

2.5.8 Descripción de caso de uso. [ST07] Enviar invitaciones

Identificador:	ST07
Nombre:	Enviar invitaciones
Resumen:	Se envían invitaciones a los usuarios seleccionados
Actores:	Usuario registrado

Precondiciones:	Se ha seleccionado un tipo de historia Se ha seleccionado un género Se han seleccionado los contactos a invitar
Postcondiciones:	El usuario destino de la invitación acepta o rechaza la invitación
Flujo normal:	1. El sistema envía información al servidor para que éste envíe a los usuarios las invitaciones 2. El servidor envía las invitaciones a los usuarios indicados
Flujos alternativos:	1. Hay un error al enviar la información y se muestra un mensaje.
Inclusiones:	
Extensiones:	

2.5.9 Descripción de caso de uso. [ST08] Aceptar/Rechazar invitación

Identificador:	ST08
Nombre:	Aceptar/Rechazar invitación
Resumen:	El usuario recibe una invitación y la acepta o la rechaza
Actores:	Usuario registrado
Precondiciones:	El usuario no está escribiendo una historia en ese momento El usuario ha recibido una invitación a escribir una historia
Postcondiciones:	El usuario ha aceptado/rechazado la invitación
Flujo normal:	1. El usuario recibe un mensaje de invitación 2. El usuario acepta/rechaza la invitación 3. El sistema envía la respuesta del usuario al servidor 4. El servidor devuelve la respuesta al usuario creador de la historia
Flujos alternativos:	1. Se produce un error al enviar la información entre usuarios y servidor. Se muestra un mensaje.
Inclusiones:	
Extensiones:	

2.5.10 Descripción de caso de uso. [ST09] Escribir historia

Identificador:	ST09
Nombre:	Escribir historia
Resumen:	Los usuarios que aceptaron las invitaciones y el usuario creador de la historia escriben la historia por turnos
Actores:	Usuario registrado
Precondiciones:	Existen usuarios suficientes para crear la historia
Postcondiciones:	Se ha creado la historia y se puede almacenar en la base de datos
Flujo normal:	1. El usuario creador escribe la primera frase o pasa turno 2. El texto se envía al servidor 3. El servidor envía el texto a todos los participantes en la creación de la historia

	4. El siguiente usuario escribe o pasa turno
Flujos alternativos:	
Inclusiones:	
Extensiones:	

2.5.11 Descripción de caso de uso. [ST10] Guardar historia

Identificador:	ST10
Nombre:	Guardar historia
Resumen:	Una vez se acaba de escribir la historia, el usuario puede guardarla en la base de datos local pulsando el botón de guardar.
Actores:	Usuario registrado
Precondiciones:	Se ha dado por finalizada la historia por parte del usuario
Postcondiciones:	La historia queda almacenada en la base de datos local
Flujo normal:	1. El usuario pulsa el botón de guardar 2. La historia se almacena en la tabla de historias de la base de datos local
Flujos alternativos:	1. Se produce un error al guardar la historia. Se muestra un mensaje de error.
Inclusiones:	
Extensiones:	

3. Diseño técnico

3.1 Arquitectura global

La visión de la arquitectura global del sistema pretende dar una primera visión del conjunto de los componentes implicados en la implementación final de la solución. Para lograr una mayor comprensión del modelo pretendido, se van a detallar distintas vistas del sistema:

- ***La arquitectura física:*** Se indicarán como están distribuidos físicamente los distintos nodos que forman parte del sistema
- ***La arquitectura lógica:*** Indica los componentes conceptuales desarrollados para dar cobertura a las necesidades propuestas
- ***La arquitectura técnica:*** Indica los componentes creados y su relación con los anteriores.

3.2 Arquitectura física

El modelo utilizado se basa en la arquitectura cliente servidor. Los clientes son los usuarios que tienen instalada la app en sus teléfonos móviles y el servidor durante el desarrollo ha sido una máquina local. El esquema es el siguiente:

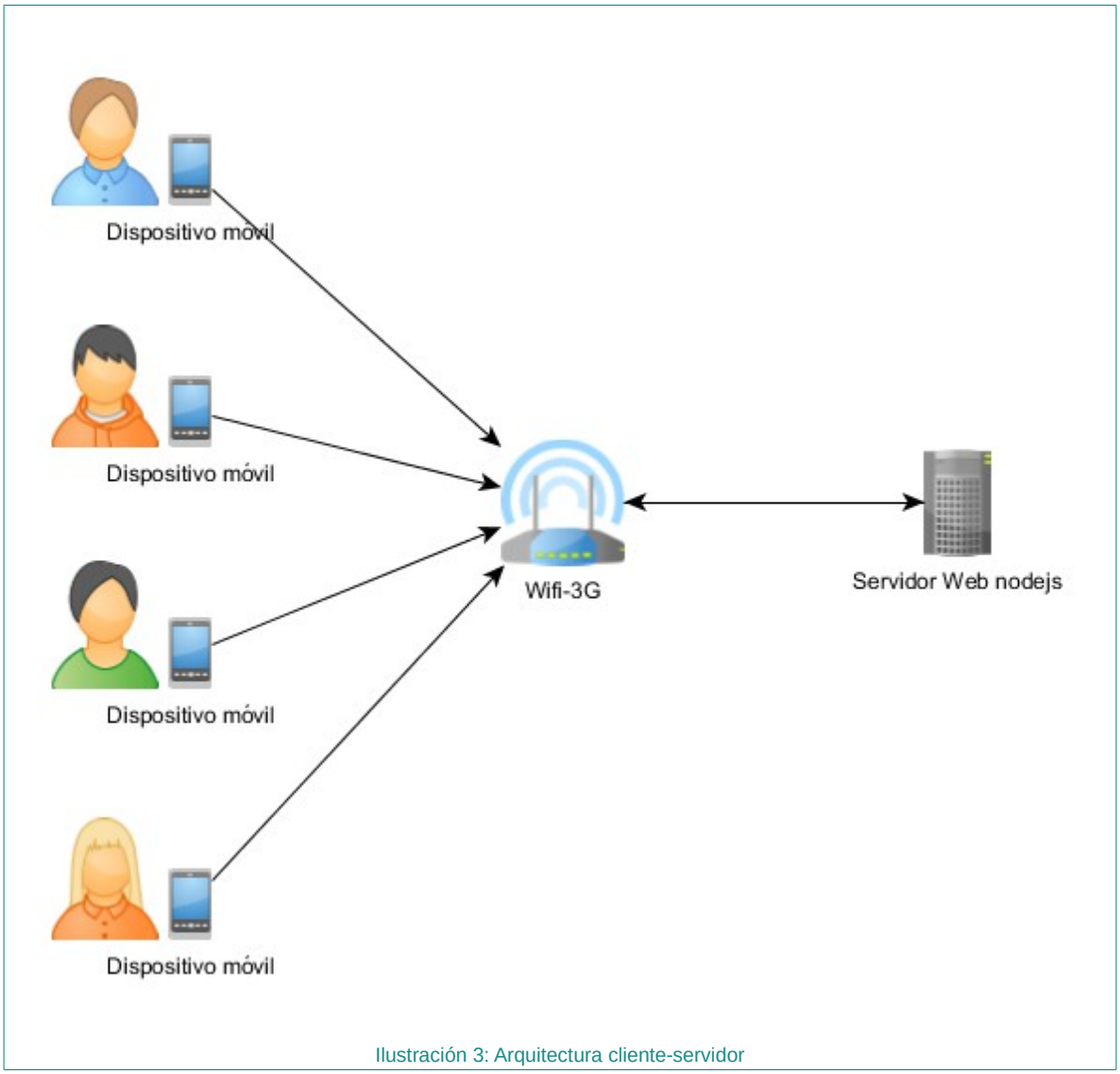


Ilustración 3: Arquitectura cliente-servidor

Como se puede apreciar en el gráfico, existe un servidor web programado en node.js que recibe las peticiones de los usuarios clientes. Este servidor contiene la lógica para la comunicación entre los usuarios al crear las historias y el acceso a la base de datos que almacena los usuarios que están conectados en ese momento.

Una vez se termina la fase de desarrollo y se finalizan las pruebas el servidor se publica en Internet para el posterior uso de la aplicación en producción.

3.3 Arquitectura lógica

La arquitectura de la aplicación móvil se basa en un único fichero HTML que contiene las distintas pantallas de la aplicación. Cada una de las pantallas que tienen alguna funcionalidad tienen un fichero JavaScript que contiene la funcionalidades asociadas a esa pantalla en concreto. Cada pantalla tiene asociada un identificador que además es el nombre del fichero JavaScript encargado de la lógica de la pantalla.

3.3.1 Arquitectura lógica del cliente

La arquitectura del cliente queda dividida en *presentación* y *funcionalidad*. La primera es la encargada de mostrar la interfaz gráfica al usuario con la que éste interactúa con la aplicación. La segunda se encarga de dar respuesta a las acciones del usuario.

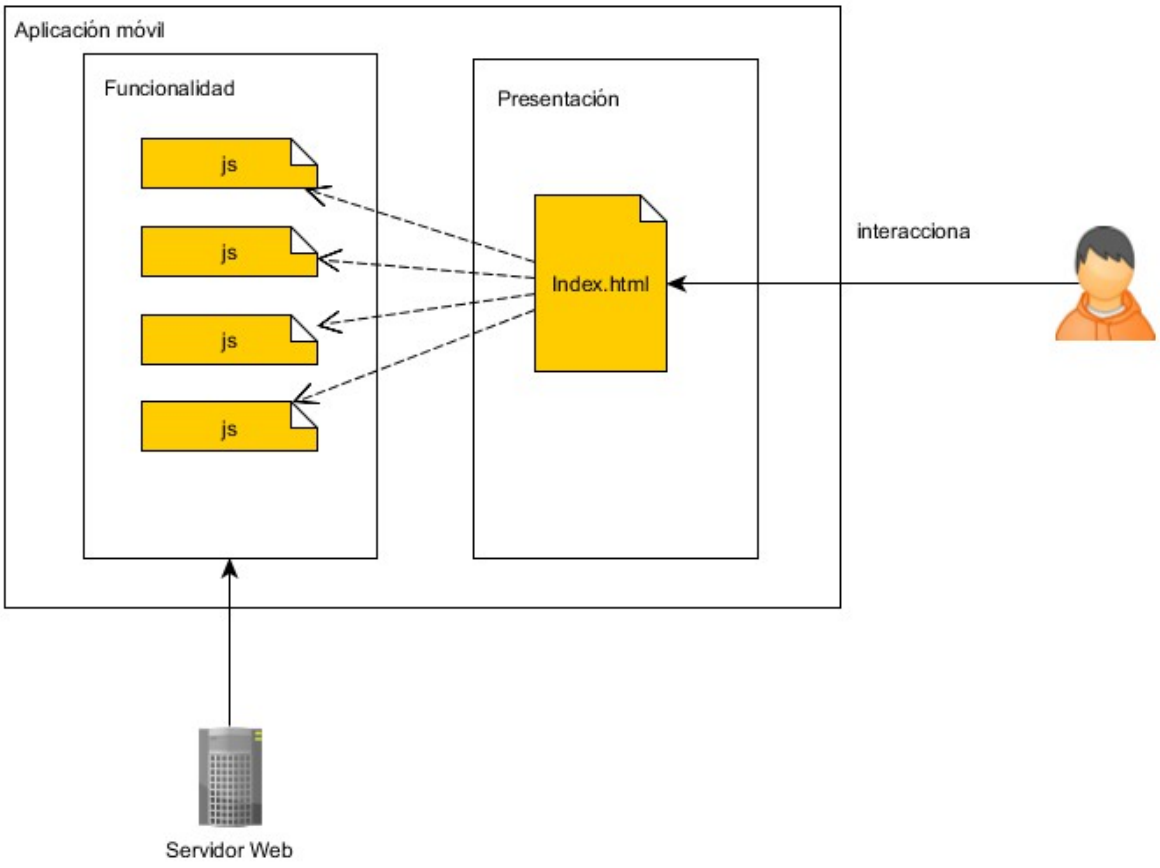


Ilustración 4: Arquitectura lógica del cliente

Presentación: Contiene el fichero HTML dividido en secciones. Cada sección es una pantalla de la aplicación. En la presentación se incluyen los ficheros CSS que dan estilo a las diferentes pantallas. Es la parte de la aplicación que muestra la interfaz gráfica que interacciona con los usuarios (botones, listados, textos etc..).

Funcionalidad: Contiene los ficheros JavaScript encargados de capturar los eventos que suceden cuando el usuario interacciona con la interfaz gráfica de la aplicación y responder con la funcionalidad asociada a ese evento. Cada pantalla contiene un fichero js y además hay un fichero js para el acceso a la base de datos local que es utilizado en varios de los anteriores.

3.3.2 Arquitectura lógica del servidor

En el servidor se ha utilizado node.js, que utiliza una arquitectura orientada a eventos. Por lo tanto, se ha seguido esa filosofía, asociando funciones a los eventos que se llaman desde la aplicación móvil.

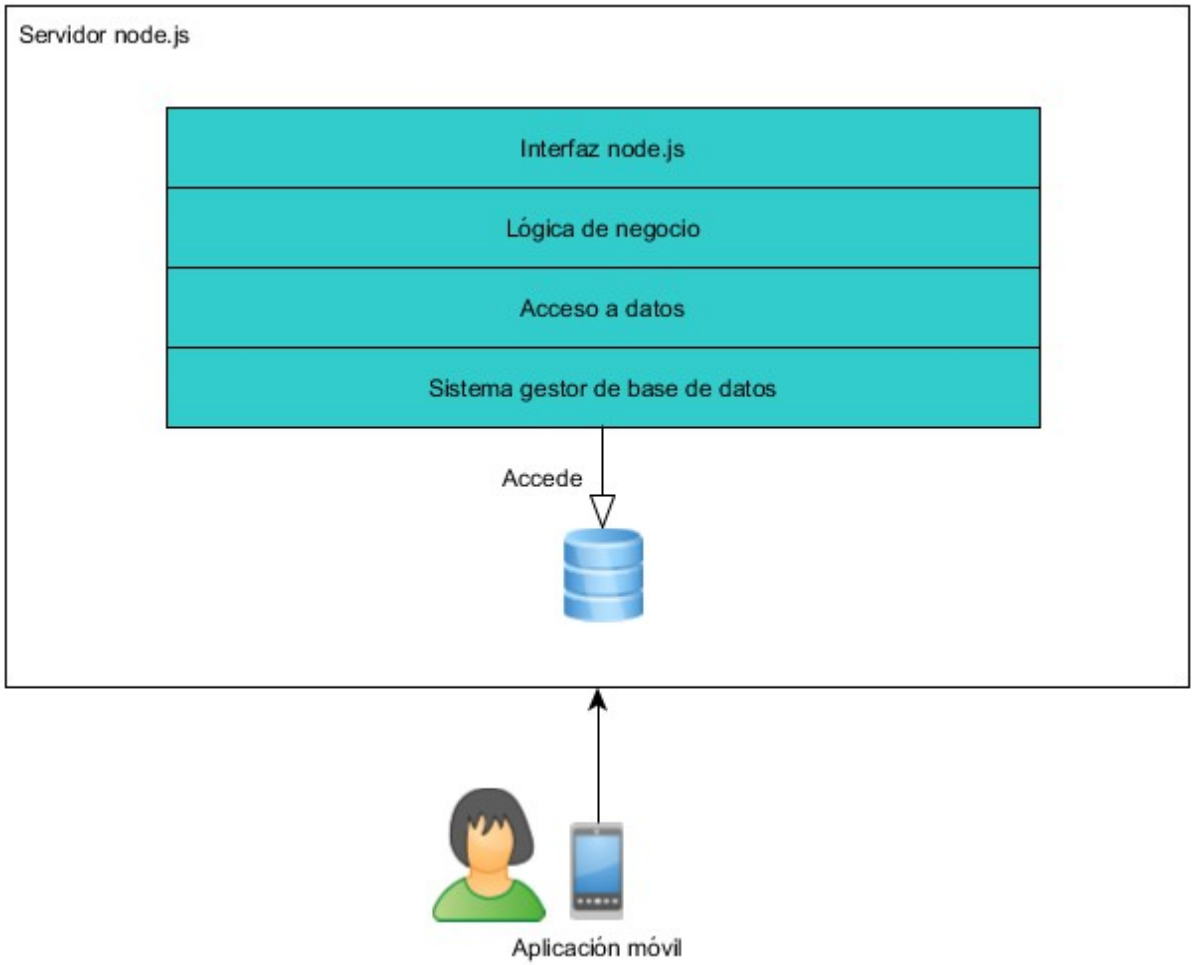


Ilustración 5: Arquitectura lógica del servidor

- ***Interfaz node.js:*** Contiene las funciones de entrada al servidor. El cliente móvil llama a estas funciones del servidor. Es el enlace entre el dispositivo móvil y la lógica de negocio del servidor.
- ***Lógica de negocio:*** Contiene las funciones encargadas de responder a los eventos que se producen en el cliente y son propagados hacia el servidor.
- ***Acceso a datos:*** Es la capa que facilita el acceso a la base de datos del servidor.
- ***Sistema gestor de base de datos:*** Gestiona la persistencia de los datos en la base de datos y ofrece los mecanismos necesarios para la recuperación de los datos y su almacenamiento.

3.3.3 Arquitectura lógica de intercambio de información

El intercambio de información entre los clientes y el servidor se realiza mediante el protocolo de comunicación *Websocket*. Esta tecnología proporciona un canal de comunicación bidireccional sobre un único socket TCP. Se adapta especialmente bien al proyecto ya que el servidor debe manejar muchos clientes conectados que a su vez envían información a otros clientes conectados, todo esto en tiempo real. La programación en servidor se apoya en el módulo socket.io que proporciona una *API* para el desarrollo con *Websocket* y que cuenta además con una gestión de *rooms* de usuarios muy útil para este proyecto.

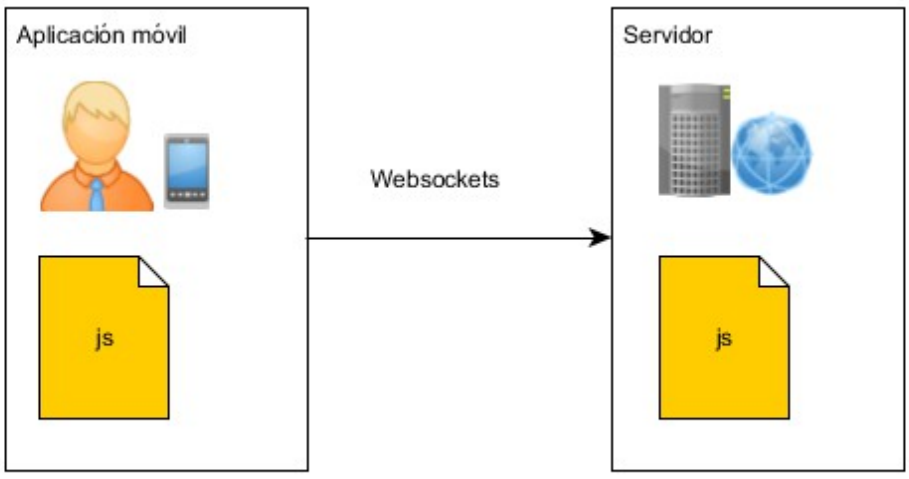


Ilustración 6: Arquitectura lógica de intercambio de información

El servidor node.js publica una serie de funciones a las cuales llama el cliente para responder a los distintos eventos que desencadena el usuario al interactuar con la aplicación.

3.4 Arquitectura de las bases de datos

3.4.1 Arquitectura de la base de datos local

La base de datos local contiene los datos del usuario que ejecuta la aplicación y de las historias que este usuario ha creado. A continuación se define el modelo de entidad-relación:

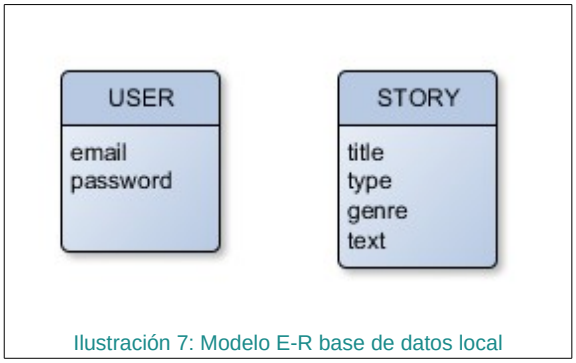


Ilustración 7: Modelo E-R base de datos local

A continuación se describe la función de las tablas:

Entidad	Descripción
USER	Contiene la información sobre el usuario que ejecuta la aplicación.
STORY	Contiene la información sobre las historias que ha creado el usuario.

Y los atributos de cada entidad:

Entidad	Atributo	Descripción
USER	Email	Email que identifica al usuario en la aplicación
USER	Password	Password introducido por el usuario
STORY	Title	Título de la historia
STORY	Type	Tipo de la historia
STORY	Genre	Género de la historia
STORY	Text	Texto que contiene el contenido de la historia

3.4.2 Arquitectura de la base de datos del servidor

La base de datos del servidor contiene los datos de los usuarios que están conectados en ese momento en la aplicación.

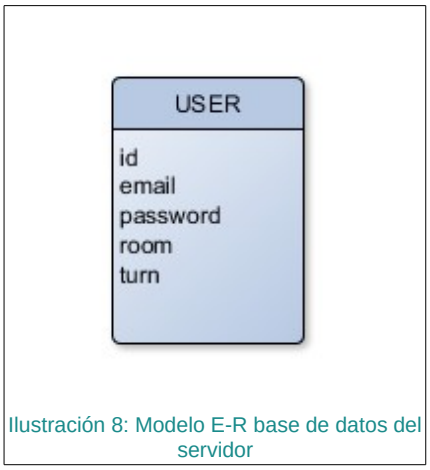


Ilustración 8: Modelo E-R base de datos del servidor

Entidad	Descripción
USER	Contiene la información sobre los usuarios conectados en ese momento en la aplicación.

Y los atributos de la entidad:

Entidad	Atributo	Descripción
USER	Id	Identificador del usuario
USER	Email	Email que identifica al usuario en la aplicación
USER	Password	Password introducido por el usuario
USER	Room	Identificador de la “sala” donde se encuentra el usuario para crear una historia
USER	Turn	Turno del usuario en el caso de que se encuentre en una sala de creación de historias

3.5 Arquitectura técnica del cliente

El desarrollo del cliente se realiza en HTML5 + CSS + JavaScript (Jquery Mobile) con Apache Cordova como framework de desarrollo para el acceso a las funciones propias del teléfono.

Así pues el diseño realizado no es un diseño clásico de clases sino un diseño orientado a obtener un prototipo, diseñando en primer lugar las diferentes pantallas correspondientes a las funcionalidades explicadas anteriormente (HTML + CSS). Como se ha comentado con anterioridad, el cliente está compuesto de un fichero principal subdividido en diferentes capas que representan las distintas pantallas de la aplicación y unos ficheros JavaScript que contienen las funcionalidades que dan respuesta a los controles de la interfaz gráfica.

A continuación se presenta un gráfico que ilustra esto:

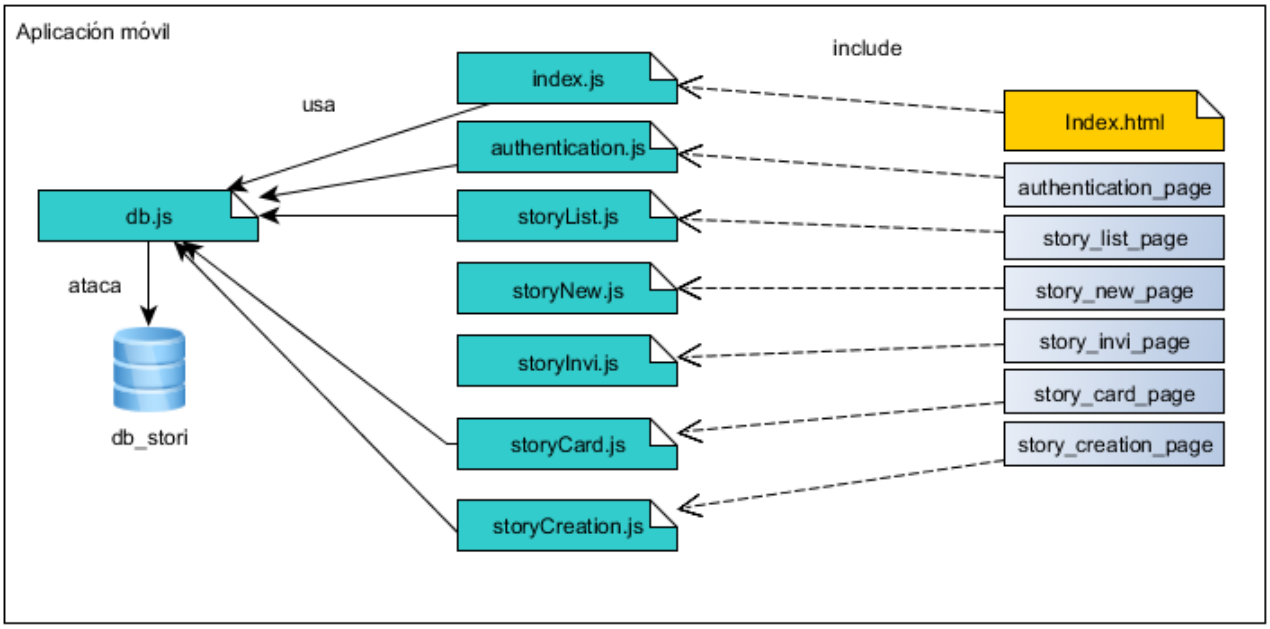


Ilustración 9: Arquitectura técnica del cliente

Como se puede observar, cada página tiene un fichero JavaScript asociado que incluye la funcionalidad de esa página. Existe un fichero JavaScript común para las funciones de acceso a la base de datos local.

A continuación se explica brevemente cada pantalla y la funcionalidad asociada a ella:

- ***Index.html:*** Se trata de la página HTML que contiene todas las pantallas de la aplicación. La subdivisión de las pantallas se ha conseguido gracias al framework de desarrollo Jquery Mobile que permite navegar entre las distintas capas o pantallas de la página HTML. La funcionalidades asociadas a esta página y que están recogidas en el fichero JavaScript index.js son aquellas relacionadas con la comunicación con el servidor node.js.

- ***Authentication_page*:** Se trata de la pantalla que muestra el formulario con los campos del email y la contraseña con el fin de que el usuario se registre la primera vez que accede a la aplicación. La funcionalidad asociada a esta pantalla se recoge en el fichero authentication.js y se encarga de la inserción de la información en la base de datos local.

- ***Story_list_page*:** Ésta es la pantalla que contiene el listado de historias que se encuentran almacenadas en la base de datos local. La funcionalidad asociada a esta pantalla se recoge en el fichero storyList.js que se encarga de consultar la base de datos para obtener las historias.

- ***Story_card_page*:** Ésta pantalla contiene los datos de una historia seleccionada a partir del listado anterior. El fichero que contiene la funcionalidad es storyCard.js y es el encargado de persistir contra la base de datos local los posibles cambios que realice el usuario sobre la historia.

- ***Story_new_page*:** Ésta pantalla se encarga de mostrar las opciones de creación de una nueva historia. Permite elegir el tipo de historia, el género y los contactos con los que escribir la nueva historia. La funcionalidad asociada a esta pantalla se encuentra en el fichero StoryNew.js y se encarga de dar funcionalidad a las acciones que realiza el usuario al interacturar con esta pantalla.

- ***Story_invi_page*:** Una vez se han elegido las opciones de creación de la historia se procede a enviar las invitaciones. Esta pantalla lista los usuarios a los que se le han enviado las invitaciones y se mantiene a la espera de su respuesta. Una vez ésta llega se procede a la creación de la historia. La funcionalidad asociada a esta pantalla se recoge en el fichero JavaScript storyInvi.js.

- ***Story_creation_page*:** Es la pantalla donde se crea la historia. Los usuarios van escribiendo por turnos hasta que alcanzan el límite de palabras y deciden guardar la historia creada en la bases de datos local o descartarla. La funcionalidad asociada a esta pantalla se encuentra en el fichero storyCreation.js.

3.6 Arquitectura técnica del servidor

El servidor es una parte fundamental del proyecto. A continuación se explicará la arquitectura diseñada, indicando los módulos requeridos para su desarrollo y explicando cada uno de los componentes. Se debe puntualizar que node.js utiliza una arquitectura orientada a eventos, por lo que no se ha realizado un diseño de clases al uso sino un diseño basado en módulos que contienen las rutinas que dan soporte a las funcionalidades necesarias.

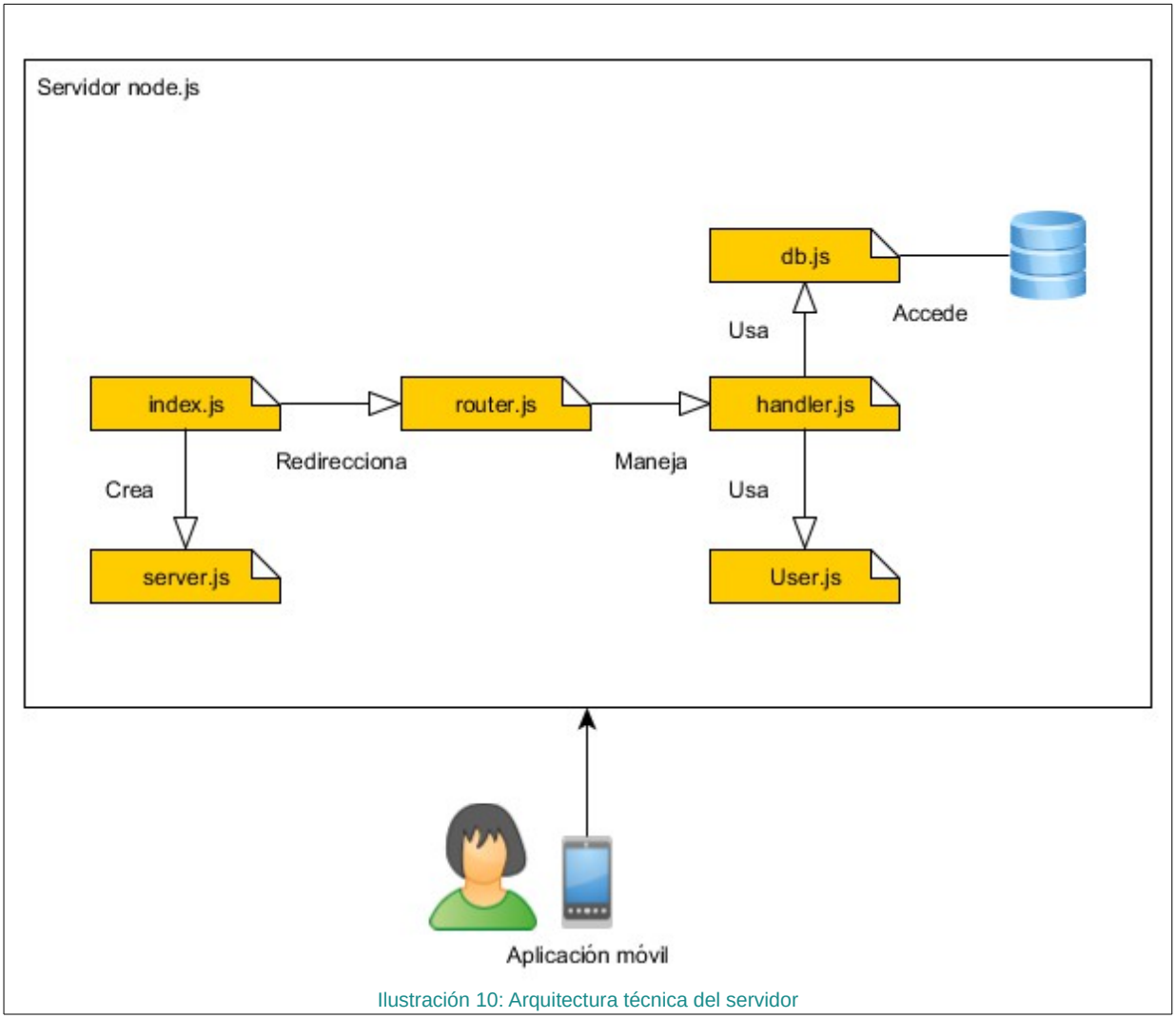


Ilustración 10: Arquitectura técnica del servidor

Como se puede apreciar en el gráfico anterior el servidor está formado por los siguientes módulos o componentes:

- ***index.js:*** Es la entrada al servidor. Este módulo define los handlers del servidor, los cuales son los encargados de responder a los eventos llamados desde el cliente. Es decir, el servidor contiene una serie de métodos los cuales pueden ser llamados por los clientes y que contienen la lógica de distintas funcionalidades.

- ***server.js:*** Se define el servidor de websockets. Este fichero contiene las funciones que pueden ser llamadas desde el cliente. Es el módulo que captura los eventos recibidos para posteriormente usando el módulo router.js que luego se explicará redirige el flujo a la funcionalidad requerida por el cliente que se encuentra en el módulo handler.js. Cabe decir que para la programación del servidor se hace uso de la tecnología de websockets. Se utiliza el módulo socket.io disponible para node.js que facilita el desarrollo del servidor.

  Socket.IO facilita la creación de apps que requieren de respuesta en tiempo real ocultando las

diferencias entre los mecanismos de los diferentes protocolos de comunicación.

- ***router.js:*** Permite controlar el flujo del servidor, es el encargado de llamar al handler correspondiente dependiendo del evento que se ha disparado. Como se ha comentado en el apartado anterior, el módulo server.js recibe una llamada a un método por parte del cliente. Para redirigir el flujo del servidor hacia la funcionalidad requerida se utiliza este módulo.

- ***handler.js:*** Contiene la lógica de negocio, las funcionalidades que responden a cada uno de los eventos. Es la parte principal del desarrollo. Hace uso tanto de la clase *User.js* como de los métodos de *db.js*.

  Es el núcleo del servidor, ya que contiene toda la lógica que da respuesta a las peticiones de los clientes. Así pues, contiene los métodos encargados de toda la funcionalidad en el servidor (Comunicación entre clientes, acceso a base de datos etc..)

- ***User.js:*** Es la clase que representa a un usuario. Se utiliza como soporte para todas las operaciones a realizar en las que intervengan usuarios.

- ***db.js:*** Contiene todas las funciones a utilizar para realizar operaciones contra la base de datos. El servidor utiliza una base de datos mongodb que almacena los usuarios que están conectados en ese momento. Para ello se ha utilizado el paquete mongojs disponible para el desarrollo con node.js que facilita las operaciones a realizar contra la base de datos.

4. Prototipo

A continuación se mostrarán las diferentes pantallas de las que consta la app, dando una breve explicación de los controles que se encuentran en ellas.

4.1 Registro/Autentificación

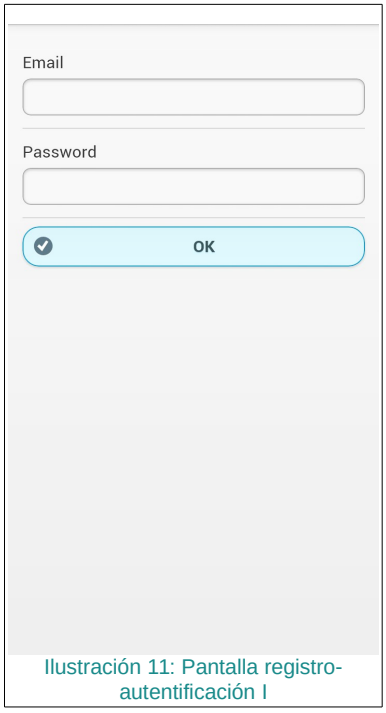


Ilustración 11: Pantalla registro-autentificación I

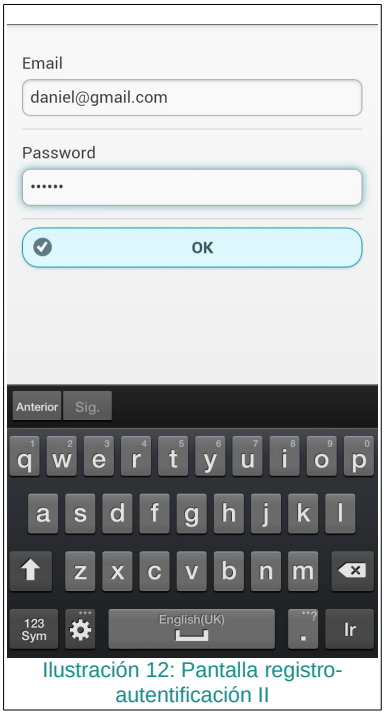


Ilustración 12: Pantalla registro-autentificación II

Las pantallas anteriores muestran como se produce el registro. La aplicación muestra un formulario y pide dos campos, el email y el password. Esta pantalla únicamente aparece en la primera ejecución de la aplicación. Cuando el usuario pulsa el botón “OK”, la aplicación muestra la pantalla principal que se muestra en el siguiente apartado.

4.2 Pantalla principal

La pantalla principal es la portada de la aplicación, se ha buscado un interfaz muy limpio, por lo tanto, solo hay un título y dos botones que a continuación se explica su funcionamiento. La pantalla es la siguiente:

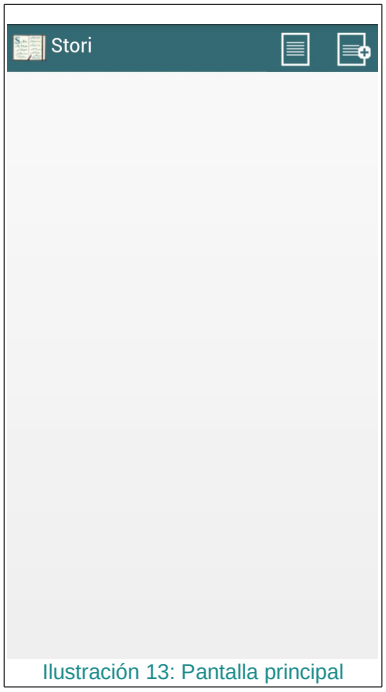


Ilustración 13: Pantalla principal

La pantalla contiene dos botones. El de más a la izquierda nos lleva al listado de las historias que el usuario tiene almacenadas en la base de datos, y el de más a la derecha nos lleva a la pantalla de creación de una nueva historia. A continuación se muestra la pantalla del listado de historias.

4.3 Listado de historias

El listado de historias muestra el título de las historias que el usuario ha ido creando y almacenando en la base de datos local

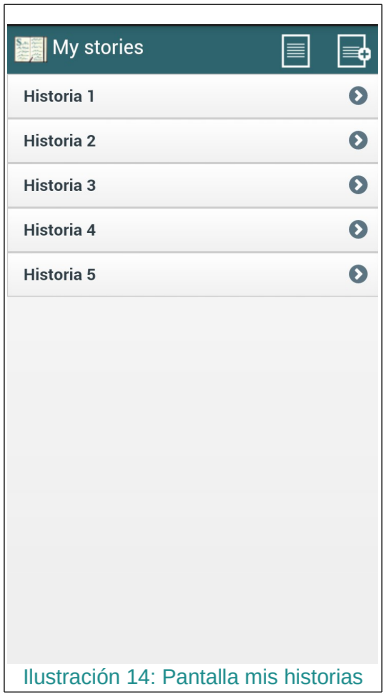


Ilustración 14: Pantalla mis historias

4.4 Ficha de la historia

Cuando el usuario selecciona en una historia del listado, la aplicación navega hasta la siguiente pantalla, en la que se muestra la información de la historia y se permite editar o eliminar.

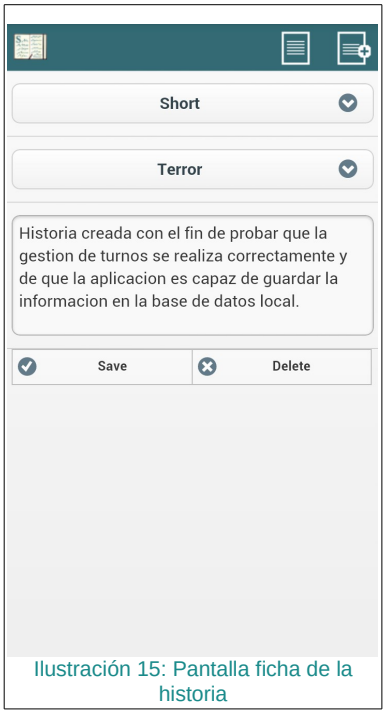


Ilustración 15: Pantalla ficha de la historia

Como se puede ver en la pantalla, hay dos controles desplegables que permiten seleccionar el tipo y genero de la historia y un control de texto que contiene la historia. Además hay dos botones, uno de ellos para guardar los cambios realizados en la historia y otro para eliminar la historia.

4.5 Crear nueva historia. Selección de tipo.

Al crear una nueva historia, lo primero es seleccionar el tipo de historia que se desea crear. La siguiente pantalla muestra como es esta selección en la aplicación. Como se puede apreciar en la pantalla, es un listado con la información del máxima de palabras por tipo a la derecha de cada uno de ellos.

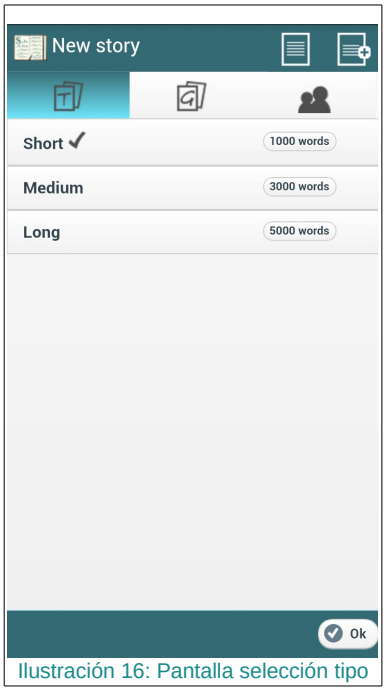


Ilustración 16: Pantalla selección tipo

4.6 Crear nueva historia. Selección de género.

Una vez seleccionado el género, se procede a seleccionar el género de la historia a crear. Para ello se muestra un listado con los géneros disponibles.

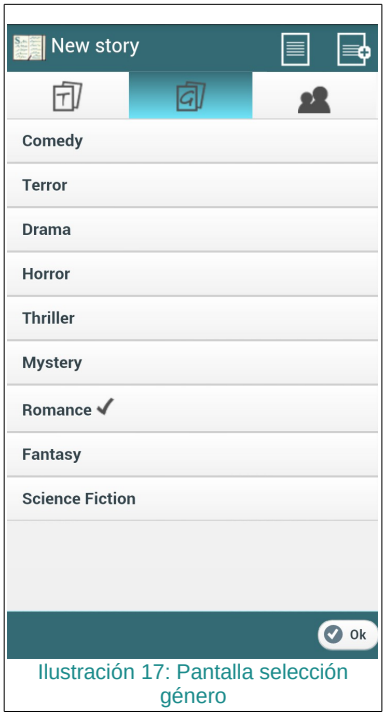


Ilustración 17: Pantalla selección género

4.7 Crear nueva historia. Selección de contactos.

Tras seleccionar el tipo y el género, se deben seleccionar los contactos con los que se desea crear la historia. La pantalla listará los contactos del teléfono, aunque para el desarrollo de la aplicación se han puesto unos contactos de prueba.

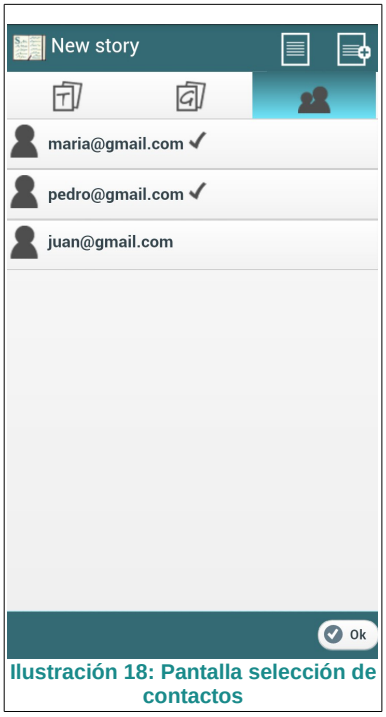


Ilustración 18: Pantalla selección de contactos

Cuando el usuario haya seleccionado todo, podrá pulsar al botón de “Ok” que le llevará a la pantalla de gestión de invitaciones, en la que se envía la petición de creación de historia a los contactos seleccionados y se espera su respuesta. A continuación se muestra la pantalla.

4.8 Gestión de invitaciones.

La pantalla de gestión de invitaciones muestra los contactos seleccionados y eventualmente la respuesta a la invitación por parte del usuario creador de la historia a participar en la creación de esta.

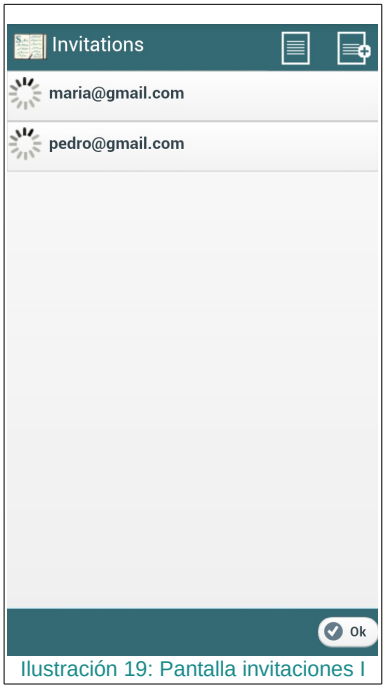


Ilustración 19: Pantalla invitaciones I

Ilustración 20: Pantalla invitaciones II

Una vez se ha obtenido respuesta de los usuarios invitados se procederá a pulsar el botón de “Ok” en para pasar a la pantalla donde se escribirá la historia.

4.9 Escribir la historia

Esta pantalla contiene la información de los contactos que están creando la historia, el turno actual, el número máximo de palabras que quedan para terminar la historia y el título de ésta

Ilustración 21: Pantalla creación

Además, como se puede apreciar, hay tres botones. Uno es para guardar la historia, otro para enviar el texto escrito y otro para pasar turno.

5. Implementación

En este apartado se comentan las decisiones que se han ido tomando durante el proceso de implementación de la aplicación y del servidor. Además, se explicará más en detalle cómo se han implementado las distintas partes tanto de la aplicación como del servidor.

5.1 Premisas de la implementación

A continuación se presentan los puntos principales que se han tenido en cuenta antes de realizar la implementación de la aplicación.

5.1.1 Código bien comentado y guías de estilo

Uno de los puntos en los que más énfasis se ha puesto es en comentar el código de tal forma que quede bien documentado que hace cada función y cada módulo dentro del proyecto. Además, se ha seguido una guía de estilo y por lo tanto el código en todas las partes de la aplicación sigue el mismo estilo para favorecer su lectura y comprensión.

5.1.2 Interfaz simple y agradable para el usuario

Otro punto principal antes de la implementación era contar con un diseño definitivo de las pantallas que fuera simple y fácil de usar a la par que agradable a la vista del usuario. Se ha invertido mucho tiempo en conseguir esto ya que se consideraba uno de los aspectos claves de la aplicación.

5.1.3 Pruebas en el navegador web

Ésta aplicación tiene la particularidad de que necesita de varios usuarios para poder crear historias. Como no se poseían varios teléfonos para poder probar, se han tenido que crear una serie de páginas HTML clientes que simulaban un usuario conectado al servidor como si fuera un usuario con la app en su móvil. Hay que tener en cuenta que el desarrollo se ha hecho sobre el terminal del desarrollador (Samsung Galaxy S3) y el no poseer otros terminales es un hándicap importante en el proceso de implementación de esta app que se ha intentado solventar con la creación de estas páginas que permiten realizar pruebas.

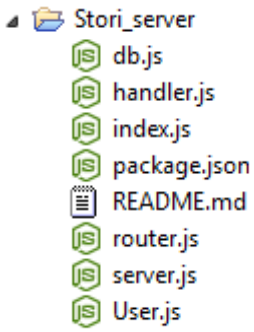
Así pues, hay que decir que el objetivo no era tener una app definitiva en un mes de implementación capaz de funcionar en todos los terminales, sino tener una app funcionando en el terminal del desarrollador y probada con las páginas que se han comentado anteriormente. Ya que el tiempo para poder tener una versión definitiva de esta app excede el tiempo disponible para realizar la implementación del TFC.

5.2 Implementación del servidor node.js

En este apartado se explicará la implementación del servidor node.js encargado de la comunicación entre los clientes de la app.

5.2.1 Organización del proyecto

La organización del proyecto del servidor es muy simple ya que sólo consta de los ficheros JavaScript correspondientes a los distintos módulos programados:



5.2.2 Index.js

Es la página principal que inicia el servidor y establece los handlers de los posibles eventos que se disparen desde los clientes:

```
var server = require("./server");
var handlers = require("./handler");
var handle = {};

handle["authenticateUser"] = handlers.authenticateUser;
handle["createStory"] = handlers.createStory;
handle["startStory"] = handlers.startStory;
handle["acceptInvitation"] = handlers.acceptInvitation;
handle["refuseInvitation"] = handlers.refuseInvitation;
handle["disconnect"] = handlers.disconnect;
handle["sendStoryMessage"] = handlers.sendStoryMessage;

server.init(handle);
```

5.2.3 Server.js

Inicia el servidor encargado de recibir las peticiones de los clientes. Cuando un cliente se conecta, se establecen los escuchadores de los eventos que éste puede disparar y cuando el cliente llama a uno de éstos se utiliza el módulo router para redirigir el flujo hacia el método que contiene la lógica de la funcionalidad requerida y que se encuentra en el módulo de handlers.

A continuación se presenta el método que establece los eventos que pueden ser llamados desde el cliente:

```
/**
 * Listen to the connection event
 */
var setEventHandlers = function() {
    io.sockets.on("connection", onSocketConnection);
};

/**
 * Listen to the events
 *
 * @param client. Client connected
 */
function onSocketConnection(client) {
    util.log("New client connected: " + client.id);

    client.on("authenticateUser", onAuthenticateUser);
    client.on("disconnect", onClientDisconnect);
    client.on("createStory", onCreateStory);
    client.on("startStory", onStartStory);
    client.on("acceptInvitation", onAcceptInvitation);
    client.on("refuseInvitation", onRefuseInvitation);
    client.on("sendStoryMessage", onSendStoryMessage);
}
```

Como se puede apreciar, con la sentencia “client.on” se escucha el evento disparado por el cliente (primer parámetro) y se llama al método del servidor asociado a ese evento (segundo parámetro).

5.2.4 Router.js

Este módulo es el encargado de llevar el flujo del programa desde la entrada al servidor hasta los métodos que contienen la lógica requerida. Es utilizado en el módulo *server.js* como se ha podido ver en la explicación del mismo. A continuación se muestra el contenido de este módulo:

```
function route(handle, method, io, client, data) {
    if (typeof handle[method] === 'function') {
        switch (method) {
            case "authenticateUser":
                handle[method](client, data);
                break;
            case "createStory":
                handle[method](io, client, data);
                break;
            case "acceptInvitation":
                handle[method](io, client, data);
                break;
            case "refuseInvitation":
                handle[method](io, client, data);
                break;
            case "startStory":
                handle[method](io, data);
                break;
            case "sendStoryMessage":
                handle[method](io, data);
                break;
            case "disconnect":
                handle[method](client);
                break;
        }
    }
}
```

Como se puede ver consta de un método route que dependiendo del método que le llega como parámetro redirige el flujo del programa.

5.2.5 Handler.js

Contiene la lógica del servidor, los métodos encargados de responder a las peticiones de los clientes que se conectan. Este módulo hace uso del módulo de acceso a la base de datos y de la clase *User.js* que se explicaran en los apartados siguientes.

La lista de métodos visibles de este módulo es la siguiente:

```
exports.authenticateUser = authenticateUser;
exports.createStory = createStory;
exports.startStory = startStory;
exports.acceptInvitation = acceptInvitation;
exports.refuseInvitation = refuseInvitation;
exports.disconnect = disconnect;
exports.sendStoryMessage = sendStoryMessage;
```

A continuación se puede apreciar uno de los métodos que contiene el módulo a modo de ejemplo.

```
/**
 *
 * @param io. Socket.
 * @param client. Client that creates the story
 * @param data. Data ent.
 * @returns
 */
function createStory(io, client, data) {
    var users, genre, type, host, room, id, email;

    // Get users, genre and type
    users = data.users.split(",");
    genre = data.genre;
    type = data.type;

    // Obtain users
    getStoryUsers(users, function(story_users) {
        if (story_users !== []) {
            host = story_users[0];
            room = host.getRoom();
            id = host.getId();
            email = host.getEmail();

            db.updateTurn(host.getEmail(), 1, function(res) {
                if (res > 0) {
                    client.join(room); // Join to the room
                    // Emit room to the client
                    io.sockets.socket(id).emit("roomCreated", {room: room});
                    story_users.splice(0,1); // Remove host user
                    story_users.forEach(function (element, index, array) {
                        // Emit invitations
                        io.sockets.socket(element.getId()).emit("getInvitation",
                            {user: email, genre: genre, type: type, room: room});
                    });
                }
            });
        }
    });
}
```

5.2.6 User.js

Esta es la clase que representa al usuario que se conecta al servidor y se almacena en la base de datos mientras se encuentre conectado. Es una clase de apoyo utilizada principalmente en el módulo *handler.js*.

A continuación se muestra la definición como ejemplo de código:

```
var db = require("./db");

var User = function (emailParam, passwordParam) {
    var email = emailParam,
        password = passwordParam,
        id, room, turn;

    var getEmail = function() {
        return email;
    };

    var getPassword = function() {
        return password;
    };
```

5.2.7 Db.js

Este módulo contiene las funciones de acceso a la base de datos utilizadas tanto por la clase *User.js* como por el módulo *handler.js.*

La lista de métodos visibles de este módulo es la siguiente:

```
exports.saveUser = saveUser;
exports.findUser = findUser;
exports.getUser = getUser;
exports.getUsersByRoom = getUsersByRoom;
exports.deleteUser = deleteUser;
exports.createRoom = createRoom;
exports.updateRoom = updateRoom;
exports.updateTurn = updateTurn;
```

Además a continuación se muestra el código de un de los métodos a modo de ejemplo:

```
/**
 * Creates a room for the user passed by parameter
 *
 * @param email. User's email
 * @param callback. Function called after this function ends.
 */
var createRoom = function (email, callback) {
    var room = new db.ObjectId();   // Generation of an ID for the room

    // Get the user from the database
    getUser(email, function(user) {
        if (user === undefined) {
            callback(0);
        } else {
            if (user.room === "") {
                // If user isn't in a room, we update it with the new room ID
                updateRoom(user.email, room, function(res) {
                    if (res > 0) { callback(1); } else { callback(0); }
                });
            } else {
                db.users.find({room: user.room}, function(err, users) {
                    // If there aren't more users in that room
                    if (err || !users || users.length <= 1) {
                        updateRoom(user.email, room, function(res) {
                            if (res > 0) { callback(1); } else { callback(0); }
                        });
                    } else { callback(0); } // User is already in a room
                });
            }
        }
    });
}
```

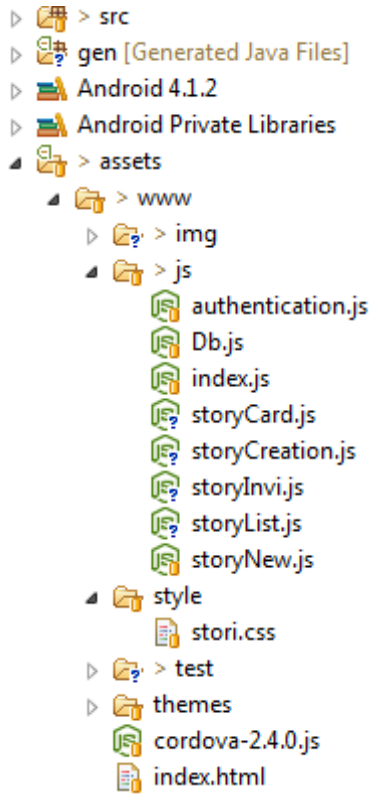
5.3 Implementación del cliente

La otra parte de la implementación es el cliente. Es la aplicación desde la cual ser escribirán las historias.

5.3.1 Organización del proyecto

La organización del proyecto cliente es muy parecida a la que se podría encontrar en cualquier proyecto web. Hay una página principal index.html y una serie de carpetas que contienen las hojas de estilos, los ficheros JavaScript, las imágenes etc...

A continuación se muestra en la imagen el árbol que estructura los principales ficheros del proyecto:



5.3.2 Index.html

Es el fichero principal de la aplicación ya que contiene todas las pantallas de ésta. Está subdividido por tags que representan las diferentes pantallas. La estructura es la típica de las páginas HTML. Primero está la cabecera encerrada entre los tags <head> que contiene los estilos y la inclusión de los ficheros JavaScript. A continuación se muestran ambos:

```
<link rel="stylesheet" href="./style/stori.css">
<link rel="stylesheet"
    href="http://code.jquery.com/mobile/1.3.1/jquery.mobile.structure-1.3.1.min.css" />
<link rel="stylesheet"
    href="http://code.jquery.com/mobile/1.3.1/jquery.mobile-1.3.1.min.css" />
<link rel="stylesheet" href="themes/Stori.min.css" />

<script type="text/javascript" charset="utf-8" src="cordova-2.4.0.js"></script>
<script src="http://code.jquery.com/jquery-1.9.1.min.js"></script>
<script
    src="http://code.jquery.com/mobile/1.3.1/jquery.mobile-1.3.1.min.js"></script>
<!-- <script src="http://stori-server.jit.su/socket.io/socket.io.js"></script>-->
<script src="http://192.168.1.3:8080/socket.io/socket.io.js"></script>

<script src="./js/Db.js"></script>
<script src="./js/index.js"></script>
<script src="./js/authentication.js"></script>
<script src="./js/storyList.js"></script>
<script src="./js/storyCard.js"></script>
<script src="./js/storyNew.js"></script>
<script src="./js/storyInvi.js"></script>
<script src="./js/storyCreation.js"></script>
```

Como se puede apreciar se incluyen los estilos propios del theme de jquery mobile creado para la aplicación y una página de estilos que contiene otras clases CSS utilizadas en los ficheros JavaScript. El resto del contenido de este fichero son las pantallas. Todas ellas se caracterizan por estar divididas en tres partes:

- ***Barra de título:*** Contiene el título de la pantalla, el icono de la aplicación y los botones del listado de historias y de crear nueva historia. En el código se diferencia porque el tag contiene el atributo *data-role = "header"*
- ***Contenido:*** Es la parte central de la pantalla. Muestra los controles propios de la pantalla actual y se diferencia del resto porque el tag contiene el atributo *data-role = "content"*
- ***Pie de página:*** Es la parte inferior de la pantalla. Puede o no contener botones que realizan alguna acción. Se diferencia porque el tag contiene el atributo *data-role = "footer"*

Como ya se ha comentado, para implementar las pantallas se han utilizado los controles que proporciona J*query Mobile*, ya que se adaptan bien a los dispositivos móviles y para la distribución de éstos en las pantallas se han utilizado los estilos CSS adecuados.

En resumen, el listado de pantallas desarrolladas son:

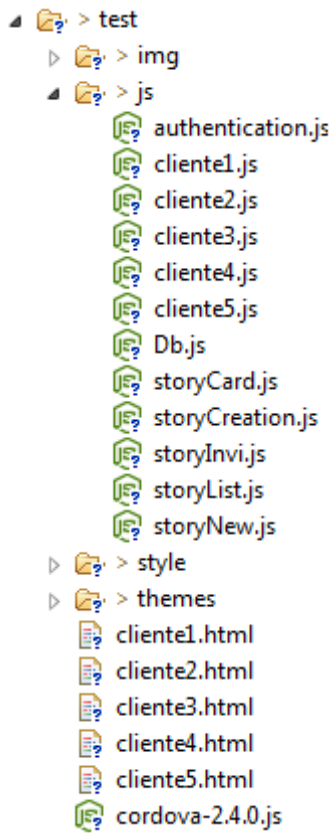
- Home_page
- Authentication_page
- My_stories_page
- Story_card_page
- Story_new_page

- Story_invi_page
- Story_card_page

5.3.3 Testing

Durante la implementación de este proyecto se han ido encontrando numerosas trabas que se han ido superando. Uno de los principales problemas ha sido el realizar las pruebas. En el proyecto se ha incluido una carpeta llamada “test” que contiene una copia de la estructura principal del proyecto pero con varias páginas “HTML” diferentes. Éstas páginas representan varios clientes y son las que han hecho posible probar la aplicación a través de un cliente web contra el servidor node.js alojado en la propia máquina.

A continuación se muestra el contenido de la carpeta test



Como se puede apreciar por cada cliente hay un fichero JavaScript con el mismo nombre que sustituiría al fichero *“index.html”* y que este adaptado para poder realizar las pruebas.

6. Funcionamiento de la aplicación

La intención de este apartado de la memoria es la de explicar brevemente el funcionamiento y el flujo de la aplicación. Así pues se presentarán las pantallas indicando cómo se desarrolla el flujo hacia otras pantallas, comunicaciones o almacenamientos.

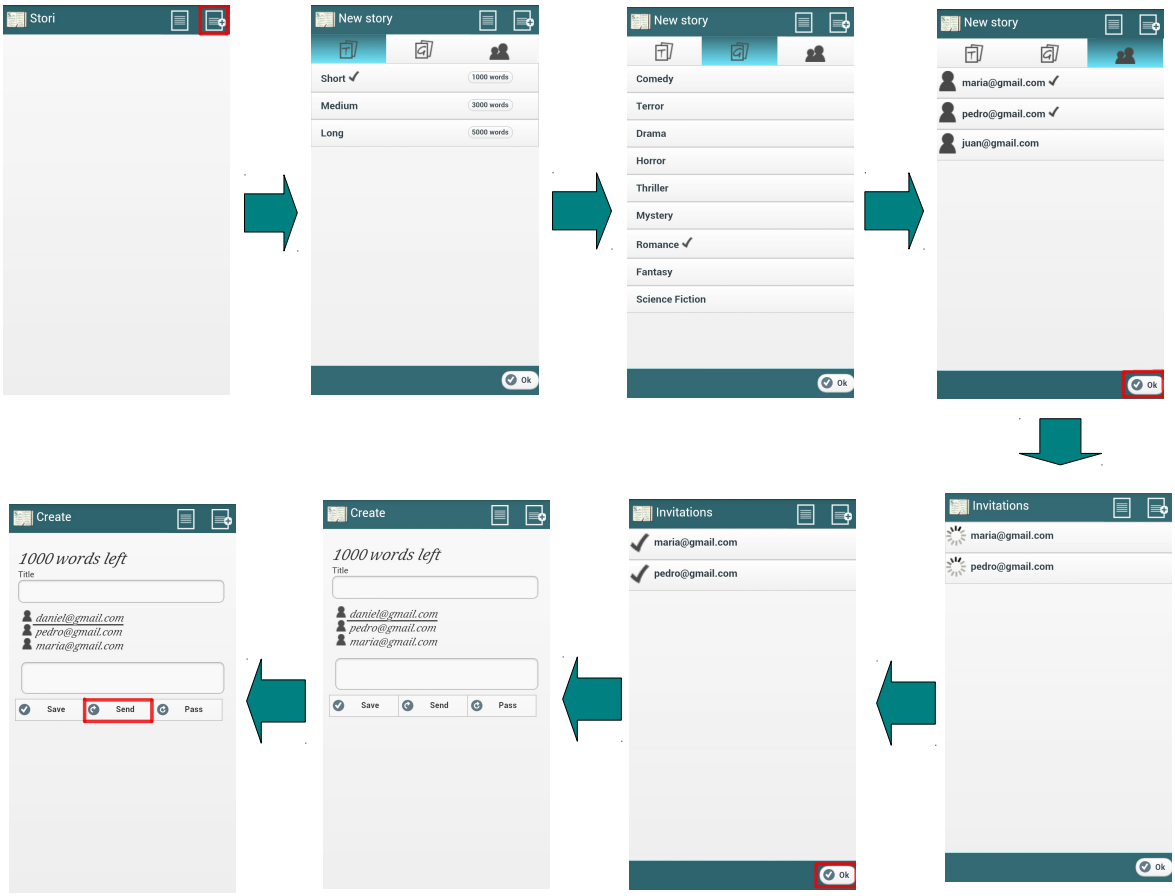
6.1 Registro / Autentificación de usuario

Como se puede ver a continuación la pantalla de registro contiene un botón “OK” que dirige el flujo del programa hacia la pantalla principal de la app.

Una vez se pulsa el botón de “OK” la aplicación guarda el usuario en la base de datos local del smartphone y redirige el flujo de la aplicación a la pantalla principal.

6.2 Creación de historia

Desde la pantalla principal pulsamos el botón de crear nueva historia. Éste nos redirige a la pantalla de creación de nueva historia en la que tenemos que seleccionar tipo de historia, género y contactos con los que queremos crearla. Una vez seleccionados pulsaremos el botón “Ok” que redirige el flujo de la aplicación hacia la pantalla de gestión de invitaciones. Una vez se obtiene respuesta sobre las peticiones de crear la historia, se pulsa “Ok” y se redirige hacia la pantalla en la que se escribirá la historia por turnos. A continuación se detalla gráficamente:



Cuando el usuario desea guardar la historia pulsa el botón “save”, el cual hace que la aplicación almacene en la base de datos local la historia con el título indicado.

6.3 Listado y actualización de historias

Para listar las historias se accederá con el botón de la página principal. Cuando se pulsa, se redirige hacia la pantalla que consulta las historias de la base de datos local y las muestra en forma de listado. En esta pantalla se puede seleccionar cualquiera del listado y esto nos llevará a la ficha de la historia que podremos modificar o eliminar. A continuación se detalla gráficamente:

7. Conclusiones

El desarrollo de este proyecto ha conllevado mucho tiempo tanto de investigación como de diseño e implementación de la solución. Aun así, se han cumplido las previsiones indicadas en el cronograma inicial. Durante la implementación se han tenido que cambiar algunas cosas y recortar algunas cosas previstas dado que el tiempo para esta fase ha sido corto para lo que se pretendía conseguir. Teniendo esto en cuenta y que la experiencia en el desarrollo de aplicaciones móviles era nula, el balance es positivo.

A continuación se explican las conclusiones sacadas de esta experiencia.

7.1 Objetivos

La mayoría de los objetivos marcados al inicio del desarrollo se han cumplido. Se ha conseguido desarrollar un servidor que soportara las funcionalidades que se requerían en el análisis previo, se ha desarrollado un aplicación móvil funcional con un diseño simple y atractivo capaz de crear historias de forma colaborativa, guardarlas, modificarlas, eliminarlas etc...

Uno de los puntos principales en los que se ha trabajado mucho es en obtener un diseño limpio y muy fácil de usar. Es uno de los puntos de los que más orgulloso estoy, ya que se ha invertido mucho tiempo en esto y el resultado es bastante satisfactorio.

Por otro lado se ha conseguido una comunicación en tiempo real entre el servidor y los clientes, que es un punto clave en este desarrollo. La programación del servidor no era ni mucho menos trivial y dado el tiempo limitado del que se disponía el resultado también es bastante satisfactorio.

En conclusión podemos afirmar que se han cumplido objetivos con solvencia, obteniendo un

producto de calidad que tras unas pruebas podría ser lanzado a un grupo de usuarios a modo de fase previa de un lanzamiento final.

7.2 Valoración personal

La experiencia de la implementación de este proyecto ha sido totalmente satisfactoria y muy gratificante. Aunque en el proceso ha habido momentos difíciles, el resumen global es muy positivo. Lo más positivo ha sido el haber sido capaz de llevar a cabo una idea compleja y materializarla en una app. Al principio tuve dudas sobre si hacía lo correcto arriesgando tanto, ya que en muy poco tiempo he tenido que aprender nuevas tecnologías y aplicarlo para que la idea fuera tomando forma, pero ahora estoy muy contento de haberlo hecho. En mi opinión ésta es una de las cosas más gratificantes de la informática. La posibilidad de imaginar algo y hacerlo realidad poco a poco es una sensación increíble.

La programación del servidor en node.js (una tecnología que desconocía) ha sido una de las mejores cosas que saco de este proyecto. El entender la arquitectura orientada a eventos, la funciones callback etc... me ha ayudado a mejorar como programador y a ampliar mis conocimientos.

El aprender a diseñar, crear los iconos, pasar tardes pensando como hacer una interfaz más clara y más fácil para el usuario pero sin sacrificar funcionalidades ha sido difícil pero muy interesante. Hasta el momento me había visto en esta situación y he de decir que quizás he invertido demasiado tiempo en esta parte del proyecto, pero me parecía un aspecto clave para la obtención de un buen resultado.

En conclusión, la valoración personal es muy positiva. Primero por haber sido capaz de plasmar una idea compleja y segundo por todo lo que he aprendido en el proceso.

7.3 Futuras mejoras

En todos los proyectos se pueden realizar mejoras y éste no es una excepción. El tiempo de desarrollo es muy pequeño y las posibilidades de expansión de este proyecto son considerables. Por eso, una de las cosas que más difíciles han resultado ha sido el ser realista y no intentar abarcar muchas cosas que harían la aplicación mucho mejor pero que materialmente era imposible realizar en el tiempo del que se disponía. A continuación se detallan las principales mejoras que se realizarán en un futuro:

- *Persistencia de los usuarios y las historias en un servidor de base de datos*: La idea inicial era que los usuarios se registraran en la app y que este usuario se guardara en la base de datos del servidor. Por motivos de seguridad y de tiempo de desarrollo esto se descartó y en su lugar cada usuario se guarda en la base de datos local del teléfono. Lo mismo pasa con las historias.
- *Incluir notificaciones en vez de alertas*: A la hora de recibir invitaciones a escribir historias, que éstas aparezcan como una notificación y no como un mensaje de alerta al usuario receptor de la invitación.

- *Hacer la app más social*: Poder realizar comentarios sobre las historias, poder calificar las historias, tener rankings de las mejores historias por tipo, género, las más votadas, las más comentadas, por fecha etc... Para esto sería necesario desarrollar un sitio web que aglutinara todas estas mejoras y modificar la app para soportarlas.
- *Creación de múltiples historias al mismo tiempo:* Permitir que un usuario escriba varias historias al mismo tiempo (con un límite por supuesto).
- *Añadir eventos propios de los móviles:* Mejorar la forma de interaccionar con el usuario, haciendo uso de los eventos que capturan los gestos realizados con los dedos sobre las pantallas táctiles.
- *Encriptación de la información que se envía al servidor:* La seguridad debería ser clave en esta app ya que se envía información sensible a través de Internet. Por lo tanto habría que encriptarla.

8. Fuentes de información

Para la realización del proyecto ha sido necesario consultar diversas fuentes de información tanto para conocer las tecnologías en las que se iba a realizar la implementación como para consultar las dudas que han ido surgiendo a medida que el proyecto avanzaba.

8.1 General

Recursos web
http://www.w3c.es/ Web para aprender sobre estándares web
http://www.ietf.org/ Web para aprender sobre estándares web
http://www.mongodb.org/ Web sobre la base de datos usada en el servidor
http://www.html5rocks.com/en/ Web para aprender sobre HTML
http://stackoverflow.com/ Web para consultas de programación
http://es.wikipedia.org/ Consultas generales

8.2 Node.js y Websockets

Recursos web
http://martinsikora.com/nodejs-and-websocket-simple-chat-tutorial.Tutorial sobre el uso de websockets
http://socket.io/. Web de la librería socket.io.
http://cometdaily.com/2011/04/06/is-websocket-chat-easier/ . Artículo sobre websockets.
http://rawkes.com/articles/creating-a-real-time-multiplayer-game-with-websockets-and-node.html. Websockets.

8.3 Cliente Móvil

Recursos web
http://jqueryui.com/ Diseño de la interfaz.
http://cordova.apache.org/ Framewor de desarrollo móvil
http://jquerymobile.com/ Framework de desarrollo para JavaScript
https://developer.mozilla.org/es/docs/JavaScript Documentación sobre JavaScript
http://msdn.microsoft.com/es-es/magazine/hh975345.aspx Artículo sobre el desarrollo con Cordova

9. Glosario

Apache Cordova	Framework para el desarrollo de aplicaciones móviles que permite a los desarrolladores acceder a las funcionalidades básicas del teléfono móvil.
Benchmark	Técnica utilizada para medir el rendimiento de un sistema.
CSS	Lenguaje de hojas de estilo utilizado para describir la presentación semántica de un documento escrito en lenguaje de marcas.
Framework	Estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, que puede servir de base para la organización y desarrollo de software.
HTML	Lenguaje de marcas para la escritura de páginas web.
JavaScript	Lenguaje dde programación interpretado. Definido como orientado a objetos, basado en prototipos, imperativo, dinámico y débilmente tipado.
Node.js	Entorno de programación en la capa del servidor basado en el lenguaje de programación JavaScript, con I/O de datos en una arquitectura orientada a eventos y basado en el motor Javascript V8.
Room	Agrupación de usuarios conectados en el servidor node.js que se disponen a crear una historia.
Socket	Concepto abstracto por el cual dos programas (posiblemente situados en computadoras distintas) pueden intercambiar cualquier flujo de datos, generalmente de manera fiable y ordenada.