1 .1 INTRODUCCIÓN

Siempre habido una gran necesidad para los robots, cuando se empezó a desarrollar sociedades casi todo el trabajo estaba realizado por grandes esfuerzos humanos y de animales. No cambio hasta la introducción de las maquinas, reduciendo drásticamente el ardor de las tareas. Al principio se utilizaban las maquinas para funciones repetitivas con maquinas simples, pero los diseños fueron avanzando para realizar tareas mas complejas.

Los primeros avances se remontan a Egipto donde el Héroe de Alexandria creo varios inventos entre ellos un motor de fuego en el siglo I DC, pasando por Leonardo Da Vinci (1452–1519) que creo varias maquinas de ingeniería como bombas hidráulicas, proyectiles de morteros, cañón de vapor…, Nikola Tesla (1856-1943) que creo un barco dirigido por control remoto y los diseños del biólogo Makto nishimura (1883-1956) que creo el robot Gakutensoku. Las maquinas fueron dando paso a la electrónica hasta llegar al primer robot digital y programable, el Unimate utilizado por General Motors en 1960 para levantar piezas de metal incandescentes.

Hoy en día los robots se parecen cada vez mas a los humanos (ASIMO, puede hablar y andar como un humano) y nos remplazan en tareas demasiados peligrosas o tediosas como son. Aunque últimamente también se esta levantando polémicas con lo que debería poder y no poder hacer un robot autónomo, se habla mucho de las tres leyes de Asimov. El desarrollo se ve muy afectado por esto ya que se intenta poner limitaciones para no remplazar completamente al humano.

No hay un acuerdo en la definición de que es un robot, pero la idea general es que es una entidad mecánica o virtual que realiza tareas automáticamente debido a su naturaleza programable. Los robots pueden ser desde humanoides (ASIMO,TOPIO) que imitan la apariencia y movimientos humanos que transmiten la sensación de intención y vida propia, hasta nano-robots, (robots cerca a la escala del nano metro) utilizados para tareas especificas especialmente en tareas militares. Pueden ser autónomos semiautónomos o controlados por control remoto

1.2 Mi Aplicación

Mi aplicación consiste en un App Android que permita de una forma segura y fiable poder controlar tu robot LEGO NXT desde tu dispositivo Android. Sera desarrollada en Java utilizando el SDK de Google en Eclipse y el brick NXT utilizara la plataforma LEJOS. La aplicación se divide en dos pasos. Primero en una autentificación de tu usuario que accede a los servicios de App Engine y segundo en la comunicación del dispositivo Android con el robot.

Con la autentificación del usuario, acedes a tu cuenta descargando tus datos e información, que permitirá comunicarse con tu robot y controlarlo desde tu dispositivo Android. La autentificación se utilizara con tu cuenta Google, gracias al API User de java para las aplicaciones App Engine, esta api permite asociar tu cuenta Google a tu robot y utilizar los servicios de almacenamiento de App Engine como el Datastore para la inteligencia artificial. Una vez conseguido tus datos podrás comunicarte con el robot a través del Bluetooth gracias a que el brick NXT lo trae incorporado.

A la par, la aplicación tendrá que hacer comprobaciones constantes en caso de fallo, para saber en que punto esta el error y si es por parte de la aplicación o de servicios externos.



.



En esta primera etapa explicare todos los conocimientos y tecnologías necesitadas para poder utilizar y desarrollar esta aplicación de una forma correcta, hare una breve descripción de cada tecnología para su mejor entendimiento

1.3 APLICACIONES ANDROID

Las aplicaciones Android son escritas utilizando el lenguaje de programación Java y utiliza como compilador las herramientas del SDK de Android. Este SDK agrupa el código, información y recursos necesitados en un paquete con la extensión .apk. Estos paquetes son usados para instalar las aplicaciones en los dispositivos Android, una vez instalados cada aplicación reside en su propio “sandbox” seguro.

Tiene varias formas de asegurar que cada aplicación este aislada, pero para entenderlo mejor primero hay que saber que del sistema operativo Android es un sistema Linux multiusuario en donde cada aplicación es un usuario distinto, de esta manera se le asigna un id de usuario único a cada aplicación. Una vez asignado el id a los archivos de la aplicación, solo la aplicación con el mismo id puede acceder a ellos. También cada aplicación tiene su propio proceso de Linux que es ejecutado por el sistema operativo y parado cuando no sea necesitado más o cuando el sistema necesite más memoria para otras aplicaciones.

Android usa el principio de menor privilegio, consiste en que cada aplicación solo tiene permiso para acceder a los recursos que necesite para hacer su trabajo. Otro método implementado es la utilización de maquinas virtuales, cada aplicación se ejecuta en su propia maquina virtual por lo que permite ejecutar código aislado de otras aplicaciones.

Aunque si hay formas de compartir información entre aplicaciones y de acceder a servicios del sistema. Esto se consigue asignando el mismo id de usuario a dos aplicaciones distintas, de esta manera pueden acceder a toda la información que tienen cada una, y para ahorrar en recursos de sistema también pueden ser ejecutadas por la misma maquina virtual. Para acceder a servicios del sistema, sms, bluetooth, contactos… se necesitan dar permisos por el usuario en el momento de la instalación.

#### 1.3.1 Componentes de una Aplicación Android

Son los bloques esenciales en la construcción de una aplicación Android, cada componente es una entrada para el sistema a tu aplicación, aunque no todos son puntos de entradas para el usuario y pueden depender unos de otros. Cada bloque es su propia entidad y tiene un rol especifico que sirve para dar forma al comportamiento general de tu aplicación. Hay 4 tipos distintos de componentes cada uno con su ciclo de vida distinto:

**Actividades**: Representa una única pantalla con el interfaz de usuario, por ejemplo una agenda de contactos tiene una actividad para añadir contactos, para leer los contactos, compartir contactos,… y aunque trabajen juntas, cada actividad es independiente de las demás. Por lo tanto otra aplicación puede empezar cualquiera de estas actividades. Por ejemplo, de un SMS se puede guardar un teléfono y crear un contacto nuevo lo que accederá a la actividad de creación de contactos directamente desde el SMS. Son implementados como una subclase de Activity

**Servicios:** Un servicio es un componente que se ejecuta de fondo no ofrece interfaz para el usuario y suelen ser operaciones que necesitan estar en ejecución constantemente o para realizar tareas para procesos remotos. Por ejemplo, un usuario puede estar utilizando una actividad a la vez que un servicio sin para la interacción entre el usuario y el dispositivo Android consiga información necesaria a través de la red. Otros componentes pueden ejecutar servicios e interactuar con ellos. Son implementados como una subclase de Services

**Proveedores de contenidos:** Son organizadores de información compartida, esta información puede estar almacenada en muchos tipos de soportes (sqlite databes, web, sd…) y a través de el otras aplicaciones puede hacer peticiones o modificaciones si el proveedor de contenidos lo permite. Un ejemplo es la agenda de contactos que tiene su proveedor de contenido el cual permite que otras aplicaciones con los permisos adecuados puedan realizar una consulta de la información de los contactos y puedan modificarlos. También sirven para leer y escribir información privada no compartida de tu aplicación. Son implementados como una subclase de ContentProviders

**Receptores de Broadcast:** Es un componente que responde a comunicados globales y toma acción. Muchos comunicados son procedentes del sistema, por ejemplo, batería baja, foto captada, pantalla apagada… pero no solo son creados por el sistema. También proceden de aplicaciones, que entre otros casos los utilizan para informar a otras aplicaciones de eventos, como la finalización de la descarga de unos archivos. Aunque no tiene interfaz de usuario si que utilizan notificaciones de estado para avisar al usuario. La idea general de estos componentes es que sirvan como una entrada para otros componentes sin realizar mucho trabajo. Son implementados como una subclase de BroadCastReciever y cada comunicado es un objeto Intent.

Como ya he mencionado un componente de una aplicación puede empezar un componente de otra aplicación, esto es un aspecto único del diseño de Android que conlleva muchas ventajas. Si necesitas que tu aplicación tome una foto puedes hacer que tu aplicación empiece el componente de la aplicación de fotos que se encarga de capturar la imagen, sin necesidad de crear una actividad nueva en tu aplicación o linquear al código de capturas de fotos. El componente propio se encargara de devolverte la foto. Además cuando el sistema lanza un componente siempre ejecuta el proceso de esa aplicación, por lo tanto, si tu aplicación necesita un componente de otra aplicación, esta no se ejecutara en el proceso de tu aplicación, se ejecutara en el suyo propio. Por esto las aplicaciones de Android no tienen una única entrada, ni tienen la función main(). Cabe mencionar que en realidad las aplicaciones no pueden lanzar directamente componentes ajenos a los suyos, tienen que enviar un mensaje al sistema Android, un intent, para que el sistema empiece ese componente. Esto es debido a que cada aplicación siempre tiene su propio proceso con permisos de archivos que restringe el acceso de aplicaciones externas.

#### 1.3.2 Activando Componentes

Los mensajes Intent son asíncronos y pertenecen a los objetos Intent. Se encargar de iniciar las actividades, servicios y los receptores de Broadcast. Pueden ser utilizados para iniciar un componente específico o un tipo de componentes especifico. En el caso de las actividades y servicios, un intent define una acción a realizar, dos casos muy comunes suelen se la de ver o enviar (view/send), para la cual necesitaran el URI(uniform resource identifier, un string de caracteres que sirve para identificar un recurso**)** de la información con la que actúan. También puedes empezar y terminar una actividad con un intent, de esta manera recibes un intent al finalizar la actividad y te permite trabajar con el al recibir su URI.

Con los receptores de broadcast, los intent solo transmiten la información que se quiere anunciar, en el caso de recibir un SMS nuevo el String indicara “SMS nuevo”

Por ultimo, los proveedores de contenido no usan intents, ellos esperan una solicitud de un ContentResolver. Los ContentResolver se encargan de las transacciones con los proveedores de contenido, si un componente necesita una información, el llamara a los métodos del contenResolver el cual se encargara de dialogar con los proveedores de contenidos para crear una capa de abstracción por motivos de seguridad.

**El archivo Manifest**

Antes de que el sistema Android pueda ejecutar una aplicación, él tiene que saber que existen sus componentes. Para ello tu aplicación tiene que declarar todos sus componentes en el archivo AndroidManifest.xml, que tiene que estar en la raíz del directorio de aplicaciones. No es la única utilidad del archivo manifest, además sirve para identificar los privilegios de usuario que necesite la aplicación, declara el api mínimo que puede usar la aplicación, declarar el software y hardware necesitado por la aplicación…

<http://developer.android.com/guide/basics/what-is-android.html>

<http://en.wikipedia.org/wiki/Android_software_development>

1.4 App Engine

El App Engine de Google te permite ejecutar aplicaciones web en las infraestructuras de Google, te permite crear aplicaciones fácilmente sin grandes costes, ni dificultades en el mantenimiento y con la posibilidad de ampliar el ancho de trafico y la capacidad de almacenamiento según tus necesidades. En general es una plataforma para instalar tus aplicaciones sin grandes dificultades, solo tienes que subir tu aplicación y App Engine se encarga del resto (servidores, mantenimiento…)

Te permite ofrecer tus aplicaciones desde tu propio dominio web o utilizando appspot.com que es gratuito, y también puedes restringir el acceso a tu aplicación si quieres ofrecerla solo a tu organización. Aunque se pueden programar en Java, Pitón y GO, para mi proyecto yo voy a utilizar java 6 JVM, que utiliza una interfaz de los Java Servet y también te permite usar JavaScript o Ruby.

Otra ventaja de este servicio, es la posibilidad de solo pagar lo que necesitas, no tiene ningún coste de montaje, ni pagos mensuales. Solo pagas por los recursos que asignes a la aplicación, de esta manera tu aplicación siempre se mantendrá dentro de tu presupuesto. Los recursos medidos son el almacenamiento y el ancho de banda asignado, medidos en gigabyte. No obstante el primer 1 GB de almacenamiento y el ancho de banda para ofreces el servicio a 5 millones de usuarios es completamente gratis, solo se te empieza a cobrar después de sobrepasar este limite y si tu lo permites.

Estos son varios servicios que ofrece App Engine a tus aplicaciones:

Ampliación automática y equilibrio de carga

Almacenamiento continuo con consultas, sorts y transacciones

Ofrecimiento dinámico de web, con soporte para la mayora de tecnologías web

Apis para autentificación de usuarios y el envió de email usando las cuentas de google

Tareas programadas para realizar un trabajo en un momento especifico y con intervalos regulares.

Un entorno de desarrollo local que permite similar el App Engine en tu ordenador.

#### 1.4.1 El Sandbox

Las aplicaciones se ejecutan en un entorno seguro que limita el acceso al sistema operativo subyacente. Estas limitaciones permite al App Engine distribuir peticiones a tú aplicación por múltiples servidor para poder afrontar a la demanda de trafico en todo momento. El sandbox también crea un entorno fiable al aislar tu aplicación, permitiendo que se ejecute independientemente del hardware, sistema operativo y posición física en los servidores web.

Tiene varias formas de limitar el acceso para crear un sandbox seguro, entre ellas, un ordenador solo puede conectarse con una aplicación a través de una petición HTTP/S en los puertos estándar, las aplicaciones no pueden escribir en el sistema de archivos y solo puede leer archivos subidos con código de aplicación, la aplicación solo puede utilizar el datastore o memcache para guardar sus datos y el código de la aplicación solo se ejecuta después de una petición web, una tarea encolada o una tarea programada, y tiene que responder en 60 segundos.

#### 1.4.2 Java Runtime Environment

En mi proyecto he utilizado el Java Runtime Environment para desarrollar mi aplicación, utilizando las normas del Java Servlet para interactuar con el entorno utilizando las JavaServer Pages que ofrecen una manera simplificada y dinámica de crear contenido web. El entorno incluye la plataforma Java SE Runtime ENvironment (JRE) 6 y sus librerías, el que permite utilizar cualquier característica de la librería mientras no viole ninguna restricción impuesta por el sandbox . Para acceder a los servicios ofrecidos por App Engine se utiliza las APIs normales de java como los APIs JavaMail, Java Persistence, Java Data Objects. Aunque haya utilizado el lenguaje java podría haber hecho uso de compiladores o interpretes compatible con JVM como JavaScript, Ruby, Scala…

La plataforma App Engine ofrece muchos servicios distintos, memcache, mail, fetch URL pero para mi proyecto voy a utilizar el datastore y Google Accounts.

#### 1.4.3 Datastore

Es una servicio de almacenamiento distribuido con un motor de consultas y transacciones. A la vez que el tu servidor web aumenta de tamaño condicionado por el trafico, el datastore aumenta en relación con tus datos. Tienes la posibilidad de elegir dos tipos de datastore, el primero (high-replication datastore) ofrece mayor disponibilidad pero mayor latencia, mientras que el slave/master datastore ofrece lo contrario. El datastore es distinto a cualquier otra base de datos relacional, los objetos de datos (entidades) tienen un tipo y unas propiedades para que las consultas puedan filtrar por tipo y con unas propiedades específicas. La estructura de las entidades de datos son asignadas por el código de la aplicación y tu aplicación puede acceder al datastore tanto como sea necesario para aplicar la estructura que necesite. El datastore es muy consistente y utiliza un control de concurrencias optimista, que permite realizar operaciones en una entidad sin bloquearla, permitiendo a múltiples transacciones sobre la misma entidad. Si se quiere actualizar una entidad que ya esta siendo modificada, se intenta un número de veces y si hay un conflicto de modificación, la actualización se deshace. También permite realizar varias operaciones en tu datastore con solo una transacción lo que te asegura que funcionen todas o ninguna.

Para poder efectuar transacciones a través de su red distribuida, utiliza grupos de entidades, asignados en su creación. Las entidades del mismo grupo se almacenan juntas para asegurar transacciones eficientes ya que cada transacción siempre trabaja con entidades del mismo grupo.

#### 1.4.4 Cuentas de Google.

He usado las cuentas de google en la autentificación de usuarios para acceder a tus datos he inteligencia artificial. De esta manera la aplicación te permite acceder con tu usuario y contraseña de google. La principal ventaja es la rapidez y accesibilidad que facilita, sin necesidad de registrase. Todos lo que quieran usar la aplicación tienen una cuenta google ya que es necesario para los dispositivos Android. También me ahorro tener que desarrollar e implementar un sistema de usuarios para mi aplicación que me permita diferenciar entre administradores y usuarios.

<http://code.google.com/intl/en/appengine/docs/java/overview.html> <http://code.google.com/intl/en/appengine/docs/whatisgoogleappengine.html>

1.5 Lego NXT

En Mi proyecto he utilizado el kit de Lego Mindstorms NXT concretamente el brick, es un ladrillo inteligente y programable que salió al Mercado en Julio del 2006 para remplazar el de la primera generación de lego mindstorms, el Robotics Invention System. Utiliza el software NXT-G para programar y permite utilizar distintos lenguajes de programación, para mi aplicación utilice leJOS NXJ. Aunque el brick tiene entradas para cuatro sensores y 3 motores a través de cables RJ12, yo solo he necesitado utilizar su capacidad de comunicación con otros dispositivos a través de bluetooth. Para navegar el brick, se utilizar una pantalla LCD de 100x64 pixeles monocromática con 4 botones, que permite acceder a todos los menús para poder configurar el brick y ejecutar tus programas. También tiene un altavoz que reproduce avisos del sistema.

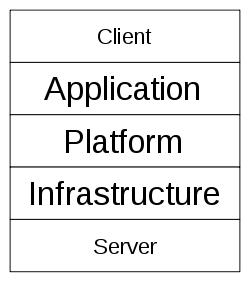
**Lejos NXJ**

# Es el firmware que utilicé para mi aplicación en el brick NXT, incluye una maquina virtual de java que permite que se programe el brick utilizando el lenguaje de programación java. Es un firmware compatible con todos los distintos tipos de sensores, desde los básicos: sensores de distancia ultrasónicos, sensores táctiles, sensores de sonido y de luz hasta los mas avanzados sensores, actuadores y multiplexores, además incluye los Apis para estos productos. Aprovechándose de la estructura orientada a objetos de java, se pueden esconder los detalles de la implementación de los sensores y actuadores tras varias interfaces ofreciendo un gran nivel de abstracción sin necesidad de preocuparse por ejemplo de los direcciones hexadecimales del hardware. Empezó como un hobby en forma de proyecto open source llamado TinyV, desarrollado por José Solórzano en 1999 hasta que se incorporaron varias personas que se pusieron a cargo y cambiaron el nombre a leJOS. Aunque siempre ha consistido de un equipo de desarrollo pequeño, tiene una gran comunidad en internet, que también participa en el desarrollo y creación de las interfaces para los sensores y actuadores. Para la comunicación bluetooth que es el aspecto esencial en la comunicación del brick con el dispositivo Android he usado las librerías Bluecove que están basadas en las librerías estándar de comunicación bluetooth iCommand.

Para el desarrollo del código leJOS he utilizado el plugin de Eclipse para ayudarme. El plugin te ayuda autocompletando/mostrando palabras reservadas, ofreciendo una estructura de pruebas y un interfaz sencillo para compilar linquear y subir al brick NXJ.

2. Arquitectura

Para mi proyecto he utilizado un servicio de Cloud computing, App Engine. Cloud computing consiste en la entrega de recursos, sowtware, computación y acceso a información/Almacenamiento en forma de servicios a través de una red y pretende ofrecer un rendimiento mayor que teniendo el software y los recursos localmente. Realiza todo esto sin que el usuario necesite saber la infraestructura o localidad de los recursos. El usuario accede a las aplicaciones a través del navegador o alguna aplicación sencilla mientras que los recursos que accedes están almacenados en servidores externos. El principal núcleo del Cloud Computing, es la posibilidad de ofrecer a empresas y usuarios la posibilidad de colgar en la red aplicaciones rápidamente ofreciendo mejor gestión y menos mantenimiento, lo que permite al servicio técnico un reajuste de recursos mas rápido y eficaz para los casos de demandas altas inesperadas.



#### 2.1 Cliente

Es software/hardware que habilita la comunicación entre el usuario y la aplicación además de la entrega de los servicios Cloud Computing. Los usuarios acceden a los servicios utilizando clientes para sus dispositivos (sobremesa, portátil, Smartphone…), por lo general estos clientes dependen de la nube completamente y son inservibles si no están conectados. En otros casos muchas aplicaciones en la nube interactúan con los navegadores de internet sin necesidad de utilizar un cliente específico para la aplicación.

#### 2.2 Aplicación

Una aplicación en la nube ofrece el software como un servicio a través de internet, eliminando la necesidad de instalar y ejecutar la aplicación en el ordenador del usuario. La entrega de este servicio se centra en el uso de paquetes de tareas interrelacionadas, estas tareas ofrecen poder de computación, almacenamiento, comunicaciones… Las tareas pueden ser clonadas en múltiples maquinas virtuales lo que permite adaptarse a las subidas y bajadas de demanda de los servicios de la aplicación. Además utilizando el archivo Configuration Data permite la modificación de parte de la aplicación sin necesidad de detenerla o relanzarla,

#### 2.3 Plataforma

Utiliza una plataforma como servicio, esta plataforma ofreciese una infraestructura en la que se crea un ambiente de abstracción y desarrollo para las aplicaciones. Gracias a su abstracción, se puede instalar aplicaciones sin tener en cuenta la complejidad de gestionar los distintos tipos de hardware y software subyacentes. También hace posible que se ofrezca los servicios del Cloud Computing y permite el desarrollo y la ejecución de aplicaciones.

#### 2.4 Infraestructura

La infraestructura de la nube, conocida como infraestructura como un servicio (IaaS) realiza una virtualización de la plataforma, esconde las características físicas de la plataforma a los usuarios, y les muestra una completamente diferente. Ofrecen memoria física y redes de comunicaciones al nivel mas básico en vez de comprar software, equipo de redes, servidores… la infraestructura permite que pagues por el servicio. Por ejemplo, si necesitas espacio para guardar datos, pagas por giga y no te preocupas de los servidores, mantenimiento e implantación.

#### 2.5 Servidores

Esta capa contiene el software y el hardware, son productos especialmente diseñados para proveer los servicios de Cloud Computing. Cada nube tiene un Software/Hardware específico, como procesadores multi-nucleo, sistema operativo, servidores… y es completamente invisible al usuario.

#### Aplicación en la nube

Tengo el código de la aplicación, ahora mismo es muy sencillo. He utilizado el api de usuarios de appengine que permite acceder a tu cuenta de google. De esta manera no necesitas registrarte en la aplicación, te identificas con tu usuario de google y no necesito programar una interfaz de loggin ya que te crea una sencilla. Necesita futuras mejoras, como el uso del datastore para guardar los datos de los usuarios. No he subido la aplicación a app engine.

#### Aplicación móvil

Para mi aplicación móvil estoy creando un app utilizando el sistema operativo Android, para el desarrollo, gracias al sdk de google, puedo ejecutarlo desde eclipse utilizando un emulador de Android para realizar las pruebas y comprobar que todo función en su ejecución. Conseguí compilar el programa sin errores y emularlo en el emulador de Android. Ahora mismo es solo una pantalla que permite escribir un mensaje y enviarlo a un brick nxj. El mensaje es solo un String y podría mejorarlo permitiendo que se enviase un objeto con más datos.

#### Aplicación robótica

Primero conseguí comunicar el brick nxj con el ordenador, tuve que instalar el software necesario siguiendo unos pasos y un cable de USB para la actualización del brick NXJ. Después escribí el programa del brick utilizando el firmware de lejos. LO escribí en Java y conseguí compilarlo a través de la pantalla de comandos y subirlo al brick. Ahora mismo solo recibe información en Strings desde el dispositivo Android y una posible mejora seria que e la comunicación fuese en ambos lados, que pudieses responder.

3 Motivación

Analizare las motivaciones de esta aplicación así como las razones por la que se necesita una aplicación como esta en el mercado de hoy y el estado del arte de los robots.

He escogido este proyecto debido a mi interés personal en la robótica y la posibilidad de trabajar con tecnologías punteras utilizando LEGO NXT Mindstorm. También quiero ofrecer un servicio que ahora mismo no existe y que si logramos desarrollar puede tener muchas aplicaciones en el campo de la robótica debido a su amplia cobertura.

Permitiría cualquier usuario conectarse con su robot desde donde quiera. Esto facilitaría mucho el desarrollo de nuevas inteligencias artificiales y robots ofreciendo una fácil solución para la comunicación entre ellos. Además hay un interés económico debido a que la robótica está cada vez más extendida y accesible al público y esta aplicación puede servir de ayuda a muchos que se estén inicializando en la robótica o incluso equipos de desarrollo.

Además quiero tener la posibilidad de trabajar con Cloud Computing que es una tecnología muy reciente, alrededor 2006, y aún más reciente es la idea de utilizarlo para ayudar en el desarrollo de robots. Esto permitirá tener toda tu información del robot accesible desde cualquier parte ofreciendo un poder de procesador y una capacidad de almacenamiento enorme.

4. METODOLOGIA

Hablare sobre la metodología seguida en el desarrollo de este proyecto tanto las reuniones de seguimiento con mi director de proyecto y las etapas de la aplicación. Analizare los objetivos que abarca esta aplicación y los objetivos que se intentara alcanzar en un futuro con posibles mejoras.

La metodología de trabajo consiste en reuniones quincenales con el director de proyecto y con un compañero que lleva otro proyecto similar en paralelo al mío. El director de proyecto puede mantener la supervisión utilizando el SVN para poder monitorizar los documentos y el estado del código. Así puede mantener un seguimiento constante e lineal de forma que el director puede actuar de forma directa e inmediata.

También se utilizan documentos de Excel para fijar tareas del proyecto y tenemos el email del director por si surge algún problema y necesita solventarse en cuanto antes. Al finalizar la fase de desarrollo se hacen pruebas del sistema, teniendo en cuenta todas las implementaciones incluso las antiguas para asegurar el funcionamiento del conjunto