UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Metodología para implementar en la nube Aplicaciones Web basadas en Java

Código: DISCA-306

Proyecto Final de Carrera

ITIS

Autor: Francisco Javier Gómez Orts

Director: Ph. D. Lenin G. Lemus Zúñiga

Septiembre, 2013

Resumen

El aumento del uso de Internet en todo el mundo y la generalización del acceso a la red exige nuevos niveles de calidad en los servicios suministrados. Los servicios ofrecidos por Internet deben ser altamente disponibles y suministrar el rendimiento esperado. Garantizar estos niveles de calidad requiere un enorme esfuerzo por parte de los proveedores y exige nuevas soluciones que se adapten dinámicamente a los niveles de carga experimentados.

Estos requerimientos cobran especial relevancia con la popularización de los dispositivos móviles, que adquieren importantes dependencias con las plataformas on-line y que no pueden admitir fallos en determinados servicios.

La nube se erige como la solución actual a estos problemas. Una infraestructura en la nube presenta un conjunto de recursos potencialmente infinitos (CPU, memoria, ancho de banda, …) que son facturados bajo demanda, conforme son utilizados. Además, el paradigma garantiza: alta fiabilidad (libre de fallos, o con tasas de fallos irrisorias) y rápida capacidad de reacción, ajustándose a los niveles de carga, en ocasiones de manera automática.

En este proyecto se describirá el concepto de nube, que servicios ofrece, tecnologías disponibles para desarrollar aplicaciones en este entorno y se explicará de forma detallada los pasos necesarios para crear una aplicación Web que funcione bajo la infraestructura de Google en la nube.

TABLA DE CONTENDOS

1. Introducción

1.1 Objetivo

Mostrar los pasos para crear una aplicación Web, que utilice los servicios de Google App Engine, para que un lector con conocimientos sobre JAVA sea capaz de: desarrollar una aplicación desde cero y hacerla completamente funcional en los servidores de Google.

1.2 El concepto de nube

Atendiendo a la definición dada por el NIST (National Institute of Standard and Technology), el cloud computing es un modelo tecnológico que permite el acceso ubicuo, adaptado y bajo demanda en red a un conjunto compartido de recursos de computación configurables compartidos (por ejemplo: redes, servidores, equipos de almacenamiento, aplicaciones y servicios), que pueden ser rápidamente aprovisionados y liberados con un esfuerzo de gestión reducido o interacción mínima con el proveedor del servicio.

Otra definición complementaria es la aportada por el RAD Lab de la Universidad de Berkeley, desde donde se explica que el cloud computing se refiere tanto a las aplicaciones entregadas como servicio a través de Internet, como el hardware y el software de los centros de datos que proporcionan estos servicios.

La idea básica es que toda la información se almacena de forma distribuida en servidores, siendo accesible en cualquier momento por el usuario sin que este se preocupe de nada, el propio sistema de “cloud” es el que se encarga de mantener siempre la información disponible. En el caso de que se esté almacenando una aplicación en la nube, el propio sistema es el que se encarga de subir la capacidad de computo, memoria, etc.. en función del uso que se le esta dando a la aplicación, con lo cual, en la nube no solo se delega la capacidad de almacenamiento sino también se distribuye en los servidores el procesamiento de datos. Esto hace que en un sistema en la

nube las capacidades de cálculo y almacenamiento sean muy elevadas. Mas tarde pasaremos a explicar los distintas implementaciones de la nube y sus servicios con mas detalle.

¿Por qué "nube"? Porque en los diagramas de flujo utilizados para representar redes, siempre a Internet se la dibuja, precisamente, como una nube, para diferenciarla de otros procesos u otras redes.

1.2.1 Características

Para poder entender de una manera rápida y sencilla cuales son las claves del concepto del cloud computing, se recurre a una serie de características principales que lo diferencian de los sistemas tradicionales de explotación de las TIC. Entre las características asociadas al cloud computing se encuentran las siguientes:

Pago por uso

Una de las características principales de las soluciones cloud es el modelo de facturación basado en el consumo, es decir, el pago que debe abonar el cliente varía en función del uso que se realiza del servicio cloud contratado.

Abstracción

Característica o capacidad de aislar los recursos informáticos contratados al proveedor de servicios cloud de los equipos informáticos del cliente. Esto se consigue gracias a la virtualización, con lo que la organización usuaria no requiere de personal dedicado al mantenimiento de la infraestructura, actualización de sistemas, pruebas y demás tareas asociadas que quedan del lado del servicio contratado.

Agilidad en la escalabilidad

Característica o capacidad consistente en aumentar o disminuir las funcionalidades ofrecidas al cliente, en función de sus necesidades puntuales sin necesidad de nuevos contratos ni penalizaciones. De la misma manera, el coste del servicio asociado se modifica también en función de las necesidades puntuales de uso de la solución. Esta característica, relacionada con el “pago por uso”, evita los riesgos

inherentes de un posible mal dimensionamiento inicial en el consumo o en la necesidad de recursos.

Multiusuario

Capacidad que otorga el cloud que permite a varios usuarios compartir los medios y recursos informáticos, permitiendo la optimización de su uso.

Autoservicio bajo demanda

Esta característica permite al usuario acceder de manera flexible a las capacidades de computación en la nube de forma automática a medida que las vaya requiriendo, sin necesidad de una interacción humana con su proveedor o proveedores de servicios cloud5.

Acceso sin restricciones

Característica consistente en la posibilidad ofrecida a los usuarios de acceder a los servicios contratados de cloud computing en cualquier lugar, en cualquier momento y con cualquier dispositivo que disponga de conexión a redes de servicio IP. El acceso a los servicios de cloud computing se realiza a través de la red, lo que facilita que distintos dispositivos, tales como teléfonos móviles, tablets u ordenadores portátiles, puedan acceder a un mismo servicio ofrecido en la red mediante mecanismos de acceso comunes.

1.2.2 Modelos de negocio

En base a la documentación analizada y tomando como referencias principales los informes del NIST (NIST Cloud Computing Standards Roadmap) y Deloitte (Cloud Computing: Forecasting change. Market Overview and Perspective) se definen tres familias fundamentales que marcan la clasificación de las soluciones cloud atendiendo al servicio que ofrecen.

Infrastructure as a Service (IaaS)

Familia de cloud computing consistente en poner a disposición del cliente el uso de la infraestructura informática (capacidad de computación, espacio de disco y bases de datos entre otros) como un servicio.

Los clientes que optan por este tipo de familia cloud en vez de adquirir o dotarse directamente de recursos como pueden ser los servidores, el espacio del centro de datos o los equipos de red optan por la externalización en busca de un ahorro en la inversión en sistemas TI.

Con esta externalización, las facturas asociadas a este tipo de servicios se calculan en base a la cantidad de recursos consumidos por el cliente, basándose así en el modelo de pago por uso.

Software as a Service (SaaS)

Familia de cloud computing consistente en la entrega de aplicaciones como servicio, siendo un modelo de despliegue de software mediante el cual el proveedor ofrece licencias de su aplicación a los clientes para su uso como un servicio bajo demanda.

Los proveedores de los servicios SaaS pueden tener instalada la aplicación en sus propios servidores Web (permitiendo a los clientes acceder, por ejemplo, mediante un navegador Web), o descargar el software en los sistemas del contratante del servicio. En este último caso, se produciría la desactivación de la aplicación una vez finalice el servicio o expire el contrato de licencia de uso.

La solución de cloud computing de Software as a Service puede estar orientada a distintos tipos de clientes según su condición:

- Usuarios particulares:

  Dropbox.

  Redes sociales.

  Gmail, Hotmail.

- Usuarios profesionales

  CRM.

  ERP.

Plataforma como un Servicio (PaaS)

El concepto conocido como Plataforma como un Servicio es básicamente un ambiente de desarrollo en donde se pueden crear otras aplicaciones que hagan uso de las características del Cloud Computing.

Un ejemplo de este servicio es Google App Engine que es el entorno en el que hemos centrado el desarrollo de la aplicación que mas adelante explicaremos.

1.2.3 Herramientas para el desarrollo de aplicaciones.

Se dispone de tres grandes herramientas para el desarrollo de aplicaciones en la nube.

1.2.3.1 Google App Engine

App Engine es un servicio de alojamiento Web que presta Google de forma gratuita, hasta determinadas cuotas y ofrecido como un servicio de cloud que permite ejecutar aplicaciones sobre la infraestructura de Google.

Actualmente las aplicaciones Google App Engine se implementan mediante los lenguajes de programación Python, Java y Go.

Se ha decidido desarrollar la aplicación con esta herramienta por los siguientes motivos:

Facilidad de inicio

App Engine es una completa pila de desarrollo que emplea tecnologías habituales para crear y alojar aplicaciones web. En App Engine, puedes crear el código de tu aplicación, probar la aplicación en tu equipo local y subirla a Google únicamente haciendo clic en un botón o introduciendo una secuencia en el símbolo del sistema. Una vez que hayas subido la

aplicación a Google, ellos se encargan de alojarla y de escalarla. Ya no tendrás que preocuparte de la administración del sistema, de la activación de instancias nuevas de la aplicación, de la fragmentación de la base de datos ni de la adquisición de equipos. Google se ocupa del mantenimiento para que tú puedas concentrarte en las funciones para los usuarios.

Escalabilidad automática

Por primera vez, tus aplicaciones pueden aprovechar las mismas tecnologías escalables sobre las que están creadas las aplicaciones de Google como, por ejemplo, BigTable y GFS. App Engine dispone de una función de escalabilidad automática, así que lo único que tienes que hacer es crear el código de tu aplicación y nosotros nos encargamos del resto. App Engine puede satisfacer tus necesidades independientemente del número de usuarios de que dispongas y de la cantidad de datos que almacene tu aplicación.

Fiabilidad, rendimiento y seguridad de la infraestructura de Google

La infraestructura de Google es famosa por su gran fiabilidad y por su alto rendimiento. Con App Engine, puedes aprovechar los diez años de experiencia que posee Google en la ejecución de sistemas escalables de forma masiva y concebidos para el rendimiento. A todas las aplicaciones App Engine les aplicamos las mismas políticas de seguridad, privacidad y protección de datos que a las demás aplicaciones de Google. Google se toma muy en serio la seguridad y dispone de medidas para proteger tu código y los datos de la aplicación.

Alojamiento rentable

El comienzo en App Engine siempre será gratuito. Puedes adquirir más recursos informáticos y pagar solo por lo que utilices.

Periodo de prueba sin riesgos

Crear una aplicación en App Engine no solo resulta fácil, sino que, además, es gratis. Puedes crear una cuenta y publicar una aplicación que se podrá utilizar inmediatamente sin ningún coste ni obligación.

Una aplicación de una cuenta gratuita dispone de hasta 1 GB de espacio y admite hasta cinco millones de vistas mensuales. Cuando estés listo para más, puedes habilitar la facturación, configurar un presupuesto diario máximo y asignarle el presupuesto a cada recurso en función de tus necesidades.

Debido a que este servicio es el elegido para desarrollar nuestra aplicación en la nube pasaremos a explicarlo con todo detalle mas adelante.

1.2.3.2 Microsoft Windows Azure

Windows Azure es una plataforma de computación en la nube y la infraestructura, creada por Microsoft, para la construcción, despliegue y gestión de aplicaciones y servicios a través de una red global de centros de datos gestionados por Microsoft. Se proporciona plataforma como servicio e infraestructura como servicios de servicios y soporta muchos lenguajes de programación diferentes, herramientas y marcos, incluyendo tanto los sistemas de software y Microsoft específicos y de terceros.

Características

- Permite a los desarrolladores crear sitios utilizando ASP.NET, PHP, o Node.js y se pueden implementar a través de FTP, Git, o Team Foundation Server.
- Las máquinas virtuales permiten a los desarrolladores migrar las aplicaciones y la infraestructura sin necesidad de cambiar el código existente, y se puede ejecutar tanto en máquinas virtuales Linux y Windows Server.
- Servicios Cloud - Plataforma de Microsoft como un entorno de servicio que se utiliza para crear aplicaciones y servicios escalables. Admite escenarios de varios niveles y despliegues automatizados.
- Gestión de datos - Base de datos SQL, anteriormente conocido como SQL Azure Database, trabaja para crear, escalar y ampliar las aplicaciones en la nube utilizando la tecnología Microsoft SQL Server. Se integra con Active Directory y Microsoft System Center y Hadoop.

- Servicios de medios de comunicación - Una oferta PaaS que se puede utilizar para la codificación, la protección de contenido, transmisión, y/o análisis.

La plataforma Windows Azure proporciona una API basada en REST, HTTP y XML que permite a los desarrolladores interactuar con los servicios prestados por Windows Azure. Microsoft también proporciona una biblioteca de clase administrada del lado del cliente que encapsula las funciones de interactuar con los servicios. También se integra con Microsoft Visual Studio, Git, y Eclipse.

Windows Azure se hizo disponible en el mercado el 1 de febrero de 2010.

1.2.3.3 Amazon Web Services

En 2006, Amazon Web Services (AWS) comenzó a proporcionar servicios de infraestructura TI para empresas en forma de servicios web, más conocido hoy como informática en nube. Se puede decir que Amazon fue pionero en ofrece este tipo de servicios en la nube.

Amazon Web Services proporciona una plataforma de infraestructura escalable de alta fiabilidad.

Se pueden desarrollar aplicaciones en los siguientes lenguajes: Java, PHP, Python, Ruby, y .NET.

1.2.4 Ventajas e inconvenientes.

Ventajas

Acceso desde cualquier sitio y con varios dispositivos. Tus programas y archivos están en la nube, con lo que te basta una conexión a Internet para acceder a ellos y usarlos de modo remoto.

Puedes hacerlo mediante un PC fijo, un laptop, un tablet PC, un iPad, un smartphone.

Todo el software está en un solo sitio. En la nube, claro está. Eso te evita tener que instalar tú los programas en tu PC, tu laptop o todos y cada uno de los múltiples equipos de una red.

Y no sólo te evita instalar el software, sino preocuparte por actualizar los programas o hacer upgrades. Tu proveedor de la nube se encarga también de eso por ti.

Casi el único programa que necesitas tener instalado es un navegador de Internet con el que acceder a la nube y trabajar en ella.
Ahorro en software y hardware. En la nube, un mismo programa lo comparten muchos usuarios, sin necesidad de tener que comprar una copia individual para cada uno de ellos. Eso abarata el precio de las aplicaciones.

Como todos esos programas se ejecutan en la nube y todo se guarda en ella, no hace falta gastar mucho dinero en un PC muy potente y con un disco duro grande.

Ahorro en mantenimiento técnico. Sin programas instalados o redes de PC complejas que configurar y mantener, los usuarios de la nube deben tener menos problemas informáticos.

El proveedor de la nube se encarga del mantenimiento técnico de sus propios servidores. El usuario no necesita saber crear redes de computadoras para compartir recursos, porque puede hacerlo a través de la nube.

Escalabilidad. Un sistema informático es escalable si puede crecer para responder a necesidades más exigentes. Esto es crucial sobre todo para las empresas.

Con la nube, la escalabilidad está garantizada sin tener que invertir más de lo necesario en previsión de que las necesidades aumenten.

Si un usuario de la nube necesita más o menos capacidad de proceso o de almacenamiento, el proveedor de la nube se lo facilitará casi en tiempo real. Eso optimiza los recursos en todo momento.

¿Seguridad? Hay una gran discusión sobre si la nube es o no más segura que los modelos tradicionales.

En principio debería serlo. Los servidores de la nube de Microsoft, por ejemplo, deben ser más seguros que mi PC o el tuyo. Pero las cosas no son tan simples. Puedes leer más abajo los detalles.

Inconvenientes de la nube

(Falta de) seguridad y privacidad. Con la computación en la nube todos tus ficheros e información pasan de estar en tu PC a almacenarse en esa nube.

Eso implica dejar de tener control sobre ellos. Nunca se puede estar seguro de quién accede a esa información o si está o no protegida como debe ser.

Eso un riesgo para usuarios particulares pero aún más para las empresas. Ellas deben confiar informaciones internas y confidenciales a un tercero, que puede o no ser fiable.

Además, es más probable que un hacker intente acceder a la nube que a un PC privado. El botín es mayor.

Sin Internet no hay nube. En la computación en la nube todo depende de que la conexión a Internet funcione. Si no es así, el cliente no podrá acceder a los programas ni los datos.

Problemas de cobertura legal. Los servidores de la nube pueden estar en cualquier parte del mundo. Si hay problemas, no está claro qué ley debe aplicarse o si ésta podrá proteger al cliente.

Conflictos de propiedad intelectual u otros. La información de los clientes ya no está en sus manos, con lo que pueden surgir problemas sobre a quién pertenece.

Eso puede llevar a situaciones delicadas, por ejemplo si el cliente pretende cambiar su proveedor de computación en la nube o si éste quiebra o comete alguna ilegalidad.

2. Google App Engine

2.1 Descripción

Google App Engine es la plataforma de Cloud Computing de Google que permite acceder a sus recursos con el objetivo de crear aplicaciones y luego hospedarlas en sus servidores. App Engine se encuentra en el nivel PaaS(Plataform as a Service), por lo que el desarrollador solamente se preocupa por desarrollar la aplicación para luego subirla a los servidores de Google abstrayéndose de demás detalles de configuración e infraestructura. Fue lanzado por primera vez como una versión beta en abril de 2008.

Dichas aplicaciones tendrán disponibilidad asegurada a partir de recursos más que suficientes para su funcionamiento, pudiendo estas almacenar y recuperar datos, hacer peticiones HTTP, enviar correos electrónicos, manipulación de imágenes y almacenamiento en caché. Las mismas pueden estar asociadas a un dominio propio o un dominio gratuito proporcionado por Google con la estructura “midominio.appspot.com”.

Se puede empezar a utilizar App Engine de forma totalmente gratuita. Todas las aplicaciones pueden utilizar hasta 500 MB de almacenamiento, suficiente CPU y ancho de banda como para permitir un servicio eficaz de la aplicación de alrededor de 5 millones de visitas a la página al mes, totalmente gratuitas. Cuando se habilita la facturación para la aplicación, se incrementan los límites gratuitos y sólo se pagan aquellos recursos que se utilicen por encima de los niveles gratuitos.

2.2 El entorno de aplicación

Google App Engine permite desarrollar fácilmente aplicaciones que se ejecuten de forma fiable, incluso con pesadas cargas de trabajo y grandes cantidades de datos. App Engine incluye las siguientes funciones:

- servidor web dinámico, totalmente compatible con las tecnologías web más comunes.

- almacenamiento permanente con funciones de consulta, clasificación y transacciones.
- escalado automático y distribución de carga.
- API para autenticar usuarios y enviar correo electrónico a través de Google Accounts,
- un completo entorno de desarrollo local que simula Google App Engine en tu equipo,
- colas de tareas que realizan trabajos fuera del ámbito de una solicitud web,
- tareas programadas para activar eventos en momentos determinados y en intervalos regulares.

Tu aplicación se puede ejecutar en uno de estos dos entornos de tiempo de ejecución: el entorno Java o el entorno Python. Cada uno de ellos proporciona protocolos estándar y tecnologías comunes para el desarrollo de aplicaciones web.

2.3 La zona de pruebas

Las aplicaciones se ejecutan en un entorno seguro que proporciona acceso limitado al sistema operativo subyacente. Estas limitaciones permiten a App Engine distribuir solicitudes web de la aplicación en varios servidores e iniciar y detener los servidores según las demandas del tráfico. La zona de pruebas aísla la aplicación en su propio entorno seguro de confianza, totalmente independiente del hardware, del sistema operativo y de la ubicación física del servidor web.

Algunos ejemplos de las limitaciones del entorno seguro de la zona de pruebas son:

Una aplicación solo podrá acceder a otros equipos de Internet a través de los servicios de correo electrónico y extracción de URL proporcionados. Otros equipos solo se podrán conectar a la aplicación mediante solicitudes HTTP (o HTTPS) en los puertos estándar.

Una aplicación no podrá escribir en el sistema de archivos. Una aplicación podrá leer archivos, pero solo aquellos subidos con el código de la

aplicación. La aplicación deberá utilizar el almacén de datos de App Engine, Memcache u otros servicios para todos los datos que permanezcan entre las solicitudes.

El código de aplicación solo se ejecuta en respuesta a una solicitud web, a una tarea en cola o a una tarea programada y debe devolver datos de respuesta en un periodo de 30 segundos en cualquier caso. Un controlador de solicitudes no podrá generar un subproceso ni ejecutar código después de haber enviado la respuesta.

2.4 El entorno de tiempo de ejecución Java

Puedes desarrollar tu aplicación para el entorno de tiempo de ejecución Java a través de herramientas de desarrollo web Java y de estándares del API conocidos. Tu aplicación interactúa con el entorno a través del estándar Java Servlet y puede utilizar tecnologías de aplicación web conocidas como, por ejemplo, JavaServer Pages (JSP).

El entorno de tiempo de ejecución Java utiliza Java 6. El SDK Java de App Engine permite desarrollar aplicaciones que utilicen tanto Java 5 como 6.

El entorno incluye la plataforma 6 de entorno de tiempo de ejecución Java (JRE) SE y bibliotecas. Las restricciones del entorno de la zona de pruebas se implementan en JVM. Una aplicación puede utilizar cualquier código de bytes de JVM o función de biblioteca, siempre que no exceda las restricciones de la zona de pruebas. Por ejemplo, si un código de bytes intenta abrir un conector o escribir en un archivo, aparece una excepción de tiempo de ejecución.

Tu aplicación accede a la mayoría de los servicios de App Engine a través de las API estándar Java. Para el almacén de datos de App Engine, el SDK Java incluye implementaciones de la interfaz de Objetos de datos Java (JDO) y de la interfaz del API de persistencia Java (JPA). Tu aplicación puede utilizar el API JavaMail para enviar mensajes de correo electrónico con el servicio de correo electrónico de App Engine. Las API HTTP java.net acceden al servicio de extracción de URL de App Engine. App Engine también incluye las API de nivel inferior para sus servicios a fin de

implementar adaptadores adicionales o para su uso directo desde la aplicación.

2.4.1 Restricciones

- Las aplicaciones solo tienen permisos de lectura a los archivos del sistema de archivos.

- Para almacenar datos y archivos en modo lectura y escritura es necesario utilizar un sistema de archivos virtual sobre el DataStore.

- Solo se puede ejecutar código a través de consultas HTTP.

- Las aplicaciones Java solo pueden usar el conjunto considerado seguro de clases del JRE estándar.

- Las aplicaciones no pueden crear nuevos hilos de ejecución.

- El soporte para SSL solo está disponible par dominios *.appspot.com.

- Un proceso iniciado en el servicio para responder a una consulta no puede durar más de treinta segundos.

- No soporta sesiones persistentes, solo sesiones replicadas a las que además se les aplican ciertos límites.

- No se pueden abrir sockets, por lo tanto, no se puede usar Twisted.

2.5 El entorno de tiempo de ejecución Python

Gracias al entorno de tiempo de ejecución Python, puedes implementar tu aplicación a través del lenguaje de programación Python y ejecutarla en un intérprete de Python optimizado. App Engine incluye varias API y herramientas para el desarrollo de aplicaciones web Python, así como un API de modelado de datos detallados, un marco de aplicaciones web fácil de utilizar, y herramientas para administrar los datos de la aplicación y acceder a ellos. También dispone de una amplia variedad de marcos y bibliotecas avanzados para el desarrollo de aplicaciones web Python como, por ejemplo, Django.

El entorno de tiempo de ejecución Python utiliza la versión 2.5.2. de Python. Google esta teniendo en cuenta una compatibilidad adicional con Python 3 para futuras versiones.

El entorno Python incluye la biblioteca estándar Python. Por supuesto, no todas las funciones de biblioteca se pueden ejecutar en el entorno de la zona de pruebas. Por ejemplo, una llamada a un método que intenta abrir un conector o escribir en un archivo generará una excepción. Para mayor comodidad, se han inhabilitado varios módulos de la biblioteca estándar cuyas funciones son incompatibles con el entorno de tiempo de ejecución y el código que los importe generará un error.

El código de aplicación escrito para el entorno Python se debe escribir exclusivamente en Python. Las extensiones escritas en lenguaje C no son compatibles.

El entorno Python proporciona varias API Python para servicios de almacén de datos, Google Accounts, extracción de URL y correo electrónico. App Engine también ofrece un sencillo marco para aplicaciones web Python denominado webapp que te permitirá empezar a crear aplicaciones fácilmente.

Se puede subir otras bibliotecas de terceros con tu aplicación, siempre que estén implementadas únicamente en Python y no requieran ningún módulo incompatible de la biblioteca estándar.

2.6 El datastore

2.6.1 Aspectos generales

El almacén de datos de App Engine ofrece un almacenamiento sólido y escalable para las aplicaciones web, con especial atención en el rendimiento de las consultas y de las operaciones de lectura. Una aplicación crea entidades donde los valores de los datos se almacenan como propiedades de una determinada entidad. La aplicación puede realizar consultas de las entidades. Todas las consultas se indexan previamente para así poder proporcionar resultados rápidos a partir de conjuntos de datos de gran volumen.

2.6.2 Introducción

App Engine ofrece dos opciones para almacenar datos que se diferencian por su disponibilidad y por su coherencia:

El **almacén de datos principal/secundario** se basa en un sistema de replicación principal-secundario que replica datos de forma asíncrona al tiempo que escribes los datos en un centro de datos físico. Dado que solo puede haber un centro de datos principal a la vez para las operaciones de escritura, esta opción ofrece consistencia fuerte para todas las consultas y operaciones de lectura. Como contrapartida, se producen periodos de inactividad temporales por incidencias en el centro de datos o por labores de mantenimiento planificadas. No obstante, la opción ofrece el coste más bajo por almacenar datos y por utilizar la CPU para tal fin.

En el **almacén de datos de replicación con alta disponibilidad,** los datos se replican en los centros de datos mediante un sistema basado en el algoritmo Paxos. Este tipo de almacén ofrece una gran disponibilidad para las operaciones de lectura y de escritura (como contrapartida, existe una mayor latencia de las operaciones de escritura). La mayoría de las consultas son de consistencia eventual. El coste de la cuota de almacenamiento y del uso de la CPU es aproximadamente tres veces superior al que supone el almacén de datos principal/secundario.

El almacén de datos de App Engine guarda objetos de datos, conocidos con el nombre de entidades. Cada entidad incluye una o varias propiedades, es decir, valores específicos de uno de los distintos tipos de datos admitidos. Por ejemplo, una propiedad puede ser una cadena, un número entero o incluso una referencia a otra entidad.

El almacén de datos puede ejecutar varias operaciones en una misma transacción. Por definición, una transacción no se lleva a cabo correctamente si alguna de las operaciones que la componen no se ejecuta. En caso de error en alguna de las operaciones, la transacción se deshace automáticamente. Esto resulta especialmente útil para aplicaciones web distribuidas, donde varios usuarios acceden a los mismos datos o los utilizan simultáneamente.

A diferencia de las bases de datos tradicionales, el almacén de datos emplea una arquitectura distribuida con el fin de administrar automáticamente el escalado para conjuntos de datos de gran volumen. La relación entre los objetos de datos propia de los almacenes de datos es muy distinta a la relación que se da en una base de datos relacional típica. Dos entidades del mismo tipo pueden tener propiedades diferentes. Se admite que varias entidades tengan propiedades con el mismo nombre pero presenten tipos de valor distintos. Si bien la interfaz del almacén de datos incorpora muchas de las funciones que integran las bases de datos tradicionales, el almacén de datos cuenta con características únicas como, por ejemplo, el diseño y la administración de los datos de forma que se pueda aprovechar la capacidad de escalado automático.

Consideración sobre el almacén de datos que se debe escoger.

El almacén de datos principal/secundario esta “deprecated” y Google recomienda usar el almacén de datos de replicación con alta disponibilidad.

2.6.3 Entidades y propiedades

Los objetos de datos del almacén de datos de App Engine se denominan entidades. Las entidades contienen una o varias propiedades, es decir, valores de uno de los distintos tipos de datos, por ejemplo, números enteros, valores de punto flotante, cadenas, fechas, datos binarios, etc.

Por su parte, cada entidad cuenta con una clave que sirve de identificador único. Las claves más simples están compuestas por un tipo y un ID de entidad que proporciona el almacén de datos. El tipo se encarga de clasificar la entidad para poder realizar consultas de esta más fácilmente. El ID de entidad también puede ser una cadena que proporcione la aplicación.

La aplicación puede extraer una entidad del almacén de datos utilizando su clave o bien realizando una consulta que coincida con las propiedades de la entidad. La consulta puede devolver entidades o no devolver ninguna. Además, los resultados pueden aparecer ordenados por los valores de propiedad. También es posible limitar el número de resultados de una

consulta con el fin de reducir el uso de memoria o agilizar el tiempo de ejecución y el uso de la CPU.

A diferencia de las bases de datos relacionales, en el almacén de datos de App Engine no es necesario que todas las entidades de un tipo determinado tengan las mismas propiedades. La aplicación puede especificar y aplicar su modelo de datos mediante las bibliotecas que se incluyen en el SDK o mediante su propio código.

Una propiedad puede incluir uno o varios valores. Si incluye varios de ellos, estos pueden ser de distintos tipos. Las consultas que se realizan a una propiedad que dispone de varios valores comprueban si alguno de estos coincide con los criterios de la consulta. Por consiguiente, estas propiedades resultan útiles para realizar comprobaciones en el caso de miembros de grupos.

2.6.4 Consultas e índices

Las consultas del almacén de datos de App Engine operan en cada una de las entidades de un determinado tipo (una clase de datos). Establecen varios filtros o ninguno para las claves y los valores de propiedad de la entidad y varios criterios de ordenación o ninguno. Si una determinada entidad incluye por lo menos un valor (posiblemente nulo) para cada una de las propiedades de los filtros y de los criterios de ordenación, y todos los valores de propiedad coinciden con los criterios de filtrado, la entidad se devuelve como resultado.

Todas las consultas del almacén de datos se basan en un índice, es decir, una tabla con los resultados de la consulta en el orden deseado. Las aplicaciones App Engine definen los índices en un archivo de configuración (si bien para algunos tipos de consulta, los índices se proporcionan de forma automática). El servidor de desarrollo web añade automáticamente sugerencias a este archivo cuando detecta consultas que todavía no tienen configurado el índice. Puedes ajustar estos índices manualmente modificando el archivo antes de subir la aplicación. A medida que la aplicación modifica las entidades del almacén de datos, este último actualiza los índices con los resultados correctos. Cuando la aplicación

ejecuta una consulta, el almacén de datos extrae los resultados directamente del índice en cuestión.

Este mecanismo admite varios tipos de consulta y sirve para la mayoría de las aplicaciones. Sin embargo, es incompatible con algunos tipos de consulta habituales en otras tecnologías de base de datos; concretamente, no se admiten consultas de unión ni de agrupación conjunta.

2.6.5 Uso del almacén de datos con alta disponibilidad

El almacén de datos de replicación con alta disponibilidad ofrece mayor disponibilidad para las operaciones de lectura y de escritura porque almacena datos de forma sincronizada en varios centros de datos. Se producen cambios en el servidor, pero no en el API. Utilizarás las mismas interfaces de programación independientemente del almacén de datos que emplees.

Sin embargo, en el almacén de datos de replicación con alta disponibilidad, las consultas que se realizan en grupos de entidades (en otras palabras, consultas que no sean de ancestros) pueden devolver resultados antiguos, esto es debido a que **este almacén de datos no garantiza la consistencia fuerte de datos**, el problema es llamado “eventually consistent". Para devolver resultados de consistencia fuerte en el entorno del almacén de datos de replicación con alta disponibilidad, la consulta debe realizarse a un único grupo de entidades. Este tipo de consulta se denomina consulta de ancestro.

Las consultas de ancestro funcionan porque los grupos de entidades son una unidad de coherencia: las operaciones se aplican al grupo en su totalidad. Las consultas de ancestro no devuelven datos hasta que el grupo de entidades completo esté actualizado. Por lo tanto, los datos que devuelven las consultas de ancestro de los grupos de entidades son de consistencia fuerte.

2.7 Uso de JPA con APP Engine

2.7.1 Aspectos generales

La interfaz del API de persistencia Java (JPA) es una interfaz estándar para almacenar objetos con datos en una base de datos relacional. El estándar permite que las interfaces realicen tareas de anotación de objetos Java, de recuperación de objetos con consultas y de interacción con una base de datos mediante transacciones. Una aplicación que emplee la interfaz de JPA puede funcionar con distintas bases de datos sin necesidad de usar el código de base de datos del proveedor. JPA simplifica la transferencia de tu aplicación entre los diferentes proveedores de bases de datos.

El SDK de Java de App Engine incluye JPA 1.0 para el almacén de datos de App Engine. La implementación se basa en DataNucleus Access Platform. JPA presenta una interfaz estándar para la interacción con bases de datos relacionales, aunque el almacén de datos de App Engine no sea una base de datos relacional. Por lo tanto, hay funciones de JPA que la implementación de App Engine, sencillamente, no admite.

2.7.2 Anotaciones soportadas

Cada objeto que guarda JPA se convierte en una entidad del almacén de datos de App Engine. El tipo de la entidad se obtiene a partir del nombre simple de la clase (sin el nombre del paquete). Cada campo persistente de la clase representa una propiedad de la entidad, con un nombre de propiedad idéntico al nombre del campo (con distinción de mayúsculas y minúsculas).

Pasamos a explicar las anotaciones de clase soportadas.

@Entity.- Para declarar una clase Java como almacenable y recuperable en el almacén de datos.

@Id.-Primary key de la entidad.

@GeneratedValue.-Indica que esa clave se auto genera.

@ManyToOne.- Indicar una relación unidireccional de uno a muchos.

@OneToMany.-Indicar una relación unidireccional de muchos a uno.

En el desarrollo de la aplicación hemos evitado usar las anotaciones @Column y @JoinColumn porque no son admitidas y para que el datastore de App Engine auto-genere el id de la entidad hemos tenido que añadir la anotación @Extension. Mas adelante pasaremos a explicar con mas detalle cómo hemos creado las entidades JPA para que funcionen en APP Engine.

2.7.3 Restricciones

- Las relaciones de propiedad multidireccionales y las relaciones sin propiedad. Puedes implementar relaciones sin propiedad mediante valores Key explícitos, aunque la comprobación del tipo no se realiza en el API.

- Consultas "Join". No puedes usar el campo de una entidad secundaria en un filtro al realizar una consulta del tipo principal. Se puede probar el campo de relación del elemento principal directamente en las consultas mediante una clave.

- Consultas de agrupación conjunta (group by, having, sum, avg, max, min).

- Consultas polimórficas. No puedes realizar consultas de una clase para obtener instancias de una subclase. Cada clase se representa mediante un tipo de entidad independiente en el almacén de datos.

3. Google Web Toolkit

3.1 Descripción

GWT o Google Web Toolkit es un framework creado por Google que permite ocultar la complejidad de varios aspectos de la tecnología AJAX. Es compatible con varios navegadores, lo cual es notorio ya que cada navegador suele necesitar código específico para lograr un front-end correcto en una aplicación web. El concepto de Google Web Toolkit es bastante sencillo, básicamente lo que se debe hacer es crear el código en Java usando cualquier IDE de Java y el compilador lo traducirá a HTML y JavaScript.

Figura 3.1: Compilador Java-JavaScript

3.2 Desarrollo

Con la biblioteca GWT, los desarrolladores pueden crear y depurar aplicaciones AJAX en lenguaje JAVA usando el entorno de desarrollo que prefieran. Cuando una aplicación es desplegada, el compilador GWT traduce la aplicación Java a un archivo Javascript, que puede ser ofuscado para optimizar el rendimiento.

GWT no es sólo una interfaz de programación, proporciona un conjunto de herramientas que permiten desarrollar funcionalidades Javascript de alto rendimiento en el navegador del cliente

Una aplicación GWT puede ser ejecutada en dos modos:

- Modo host (Hosted mode): La aplicación se ejecuta como código bytecode de Java dentro de la Máquina Virtual de Java (JVM). Este

modo es el más usado para desarrollo, soportando el cambio de código en caliente y el depurado.

- Modo web (Web mode): La aplicación se ejecuta como código Javascript y HTML puro, compilado a partir del código Java. Este modo se suele usar para el despliegue de la aplicación.

La utilidad de línea de comandos applicationCreator genera automáticamente todos los archivos necesarios para iniciar un proyecto GWT, incluso permite crear un proyecto para Eclipse.

Existen varios plugins de código abierto para ayudar a desarrollar en diferentes entornos de desarrollo, como GWT4NB para NetBeans, Cypal Studio for GWT para Eclipse o gwtDeveloper para Jdeveloper.

3.3 Arquitectura GWT

GWT contiene los siguientes componentes:

- GWT Java-to-JavaScript Compiler: la función de este componente es traducir el código desarrollado en Java al lenguaje JavaScript. Lo empleamos cuando usamos al GWT en modo web.
- Hosted Web Browser: este componente ejecuta la aplicación Java sin traducirla a JavaScript, en modo host usando la máquina virtual de Java.
- JRE Emulation Library: contiene las bibliotecas más importantes de las clases de Java: java.lang en donde se encuentran las clases fundamentales para poder programar en Java y un subconjunto de las clases del paquete java.util. Java.lang incluye, entre otras, la clase java.lang.object que es la clase fundamental de la que heredan o extienden todas las clases en Java. El resto de los paquetes no están soportados por GWT.
- GWT Web UI Class Library: contiene un conjunto de elementos de interfaz de usuario que permite la creación de objetos tales como textos, cajas de texto, imágenes y botones.

3.3.1 Patrón Modelo-Vista-Presentador

GWT usa este modelo llamado MVP en lugar del tradicional MVC.

El Patrón Modelo-Vista-Presentador (MVP) surge para ayudar a realizar pruebas automáticas de la interfaz gráfica, para ello la idea es codificar la interfaz de usuario lo más simple posible, teniendo el menor código posible, de forma que no merezca la pena probarla. En su lugar, toda la lógica de la interfaz de usuario, se hace en una clase separada (que se conoce como Presentador), que no dependa en absoluto de los componentes de la interfaz gráfica y que, por tanto, es más fácil de realizar pruebas.

La idea básica es que la clase Presentador haga de intermediario entre la Vista (la interfaz gráfica de usuario) y el modelo de datos. La vista tiene métodos en los que le pasan los datos que debe pintar ya "mascados" (una lista de cadenas por ejemplo, en vez del modelo...). Únicamente debe meter esos datos en los componentes gráficos (cajas de texto, checkbox, etc). También métodos get para obtener el contenido de esos componentes.

El Presentador hará de enlace entre el modelo y la vista, y dotará de inteligencia a la vista. Como el objetivo es poder probarlo fácilmente, el Presentador recibe las interfaces que deben implementar el modelo y la vista, con los métodos públicos a los que el Presentador debe llamar.

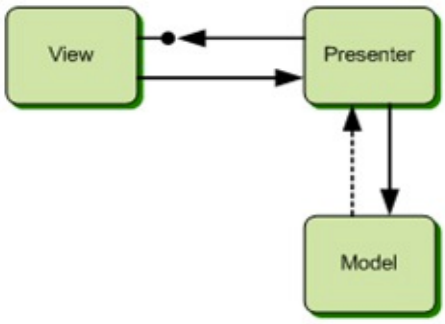

Figura 3.2: Modelo MVP

Se puede decir que el patrón MVP es una mejora del patrón Modelo-Vista-Controlador (MVC) basado en tres características:

- La vista no conoce el modelo.

- El presentador es independiente de la tecnología de interfaz de usuario.
- La vista y el presentador son testeables puesto que esta basada en un contrato.

3.4 Características

- Componentes gráficos dinámicos y reusables: los programadores pueden usar clases prediseñadas para implementar comportamientos que de otra manera consumirían mucho tiempo, como arrastrar y soltar o menús en árbol.
- Simple mecanismo RPC.
- Gestión del historial del navegador web.
- Soporte para depurado de Java.
- Control de diferentes características del navegador.
- Integración con JUnit.
- Internacionalización.
- Los desarrolladores pueden mezclar código escrito en Javascript dentro del código Java usando la Interfaz Nativa Javascript (JSNI).
- Soporte para la API de Google (inicialmente, soporte para Google Gears).
- Es de código abierto.
- Los desarrolladores pueden diseñar y desarrollar sus aplicaciones orientadas a objetos.
- Errores comunes en Javascript, como la discrepancia de tipos de datos, son controlados en tiempo de compilación.
- El código Javascript generado puede ser ofuscado para optimizar el rendimiento.
- Existen un numeroso conjunto de bibliotecas desarrolladas por Google y terceros que amplían las funcionalidades de GWT.

4. Aplicación de ejemplo

4.1 Aspectos generales

Hemos desarrollado una aplicación en App Engine cuya función es gestionar una agenda de contactos. El usuario será capaz de crear, editar y eliminar usuarios de su agenda.

Como framework de desarrollo se ha escogido GWT ya que me parece un lenguaje muy interesante por las características mencionadas en el punto 3.4, porque mejora la experiencia de usuario y además el plugin de Google App Engine para Eclipse te ofrece la opción de crear un proyecto de tipo GWT directamente con la estructura y librerías necesarias para empezar el desarrollo.

4.2 Herramientas usadas

4.2.1 Eclipse

Se ha utilizado **Eclipse Helios 3.6.2** como IDE para el desarrollo la aplicación.

Es muy fácil utilizar el entorno de desarrollo de Eclipse para desarrollar una aplicación Java de App Engine, así como cualquier otra aplicación web basada en Servlet. Con el plugin de Google para Eclipse, es aún más fácil.

4.2.2 Plugin de Google App Engine

El plugin te permite crear, probar y subir aplicaciones App Engine desde Eclipse.

El plugin de Google para Eclipse también permite desarrollar aplicaciones a través de Google Web Toolkit (GWT) para ejecutarlas en App Engine o en cualquier otro entorno.

Mas adelante explicaremos que versión hemos utilizado y como instalar el plugin en Eclipse Helios.

4.2.3 SDK de App Engine y GWT

Con el plugin de App Engine para Eclipse obtendremos también los SDK.

4.3 Diseño

Se ha creado un diseño de la aplicación sencillo, sin mucha sobrecarga de elementos, intentando que sea los mas intuitivo posible para el usuario. De esta forma aportamos una mejor interacción con la aplicación y unido al uso de la tecnología GWT conseguimos una mejor experiencia de usuario.

4.3.1 Vista principal

La vista principal de la aplicación se divide en 3 partes:

Barra superior donde aparece el logo de la aplicación, una pequeña descripción y el usuario logado en el sistema.

Barra lateral de menú que ofrece las funcionalidades disponibles en la aplicación(Crear contacto, Editar contacto, Eliminar contacto y Ver detalles);

Panel central en el que se muestra un listado de los contactos disponibles en la agenda. Este listado muestra:

- Foto del contacto, en caso de que no tenga foto añadida se muestra una imagen genérica.
- Nombre completo
- Número de teléfono, si el contacto tiene varios números de teléfono se mostrará el principal.

Imagen ilustrativa de la vista principal.

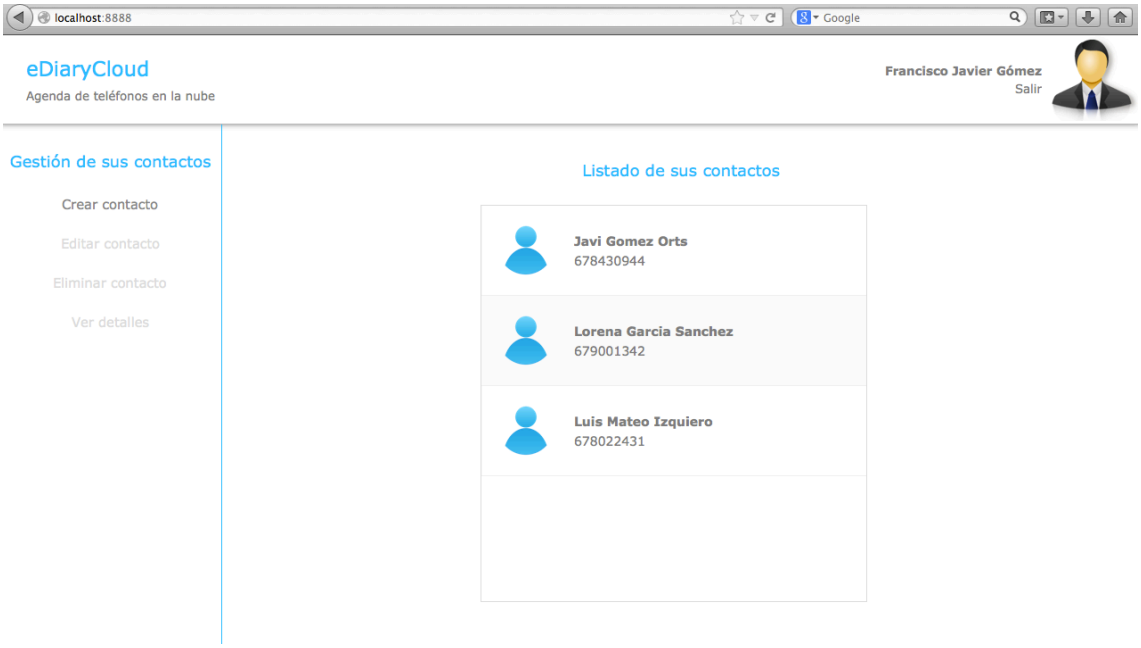

Figura 4.1

4.3.2 Vista crear contacto

Se accede pulsando en la opción de menú “Crear contacto”. Esta vista es básica un formulario donde se permite rellenar los campos contemplados para crear un contacto nuevo. Los campos contemplados son: nombre, primer apellido, segundo apellido, email y teléfono.

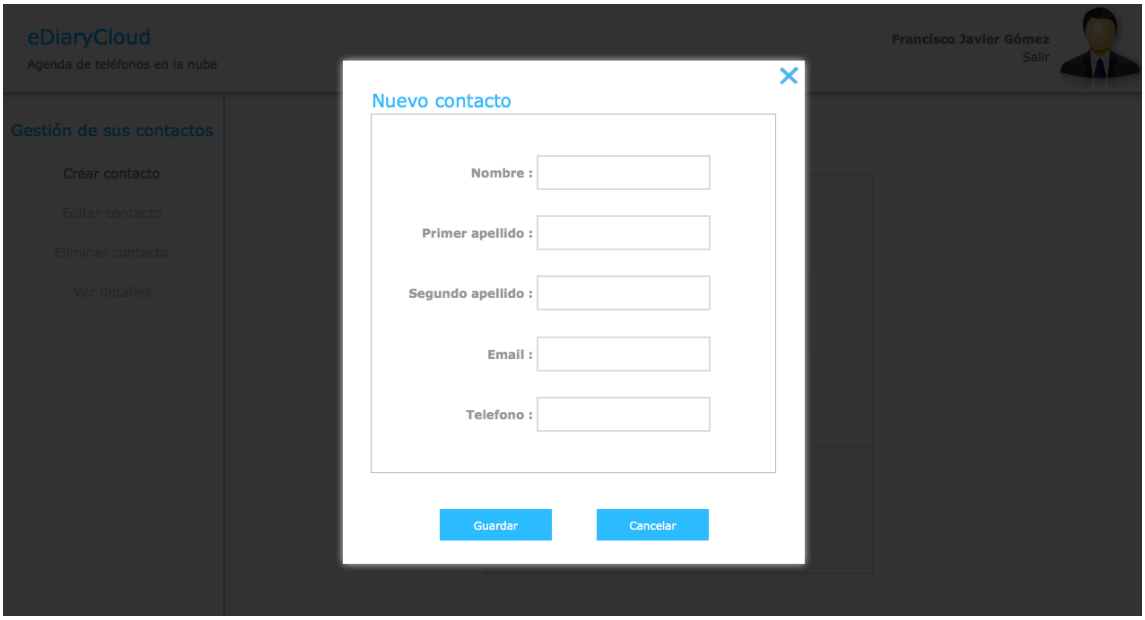

Figura 4.2

Cuando se pulsa en “Guardar” el sistema válida que todos los campos sean correctos, si hay alguno que no cumple la validación se suspende la creación del contacto y se avisa al usuario de los campos no válidos.

4.3.3 Vista editar contacto

Solo podemos acceder a esta vista si tenemos un contacto seleccionado en el listado, de lo contrario la opción de menú asociada a esta vista aparece desactivada.

Se encarga de precargar la información del contacto en un formulario editable.

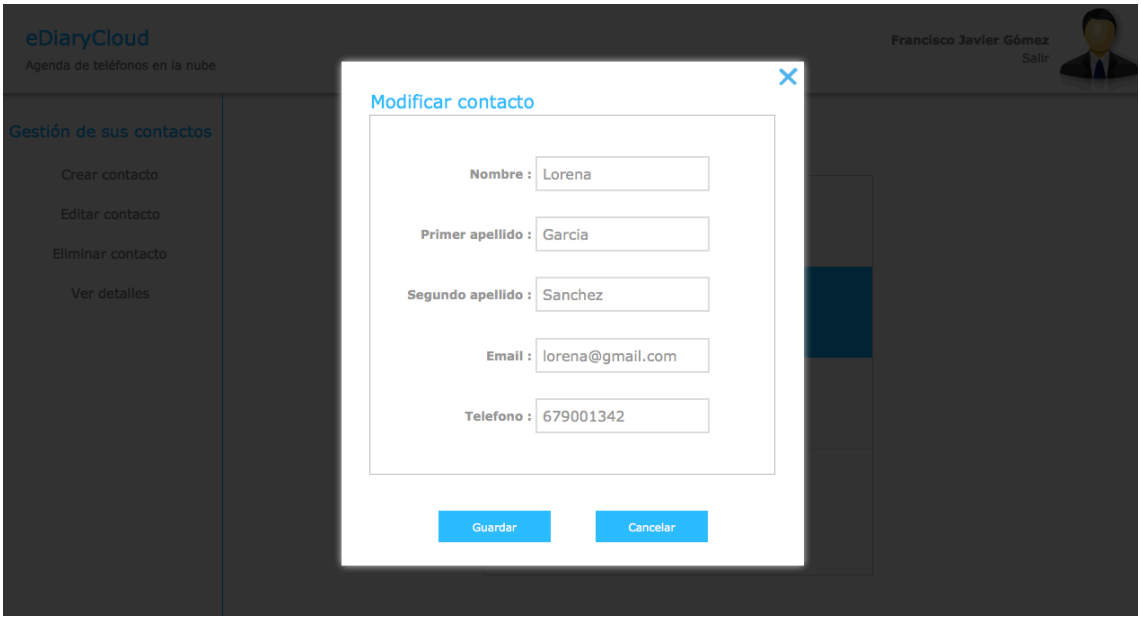

Figura 4.3

El usuario puede modificar los campos que desea y pulsar en “Guardar”.

4.3.4 Vista eliminar contacto

También aparece desactivada la opción de menú para acceder a la vista hasta que no se tenga seleccionado un contacto a eliminar.

Cuando se selecciona un contacto y se pulsa en eliminar el sistema pregunta si realmente se desea eliminar a ese contacto. En caso afirmativo se procede a la eliminación y si se cancela el proceso se vuelve a la vista principal.

Imagen ilustrativa sobre el proceso de eliminación.

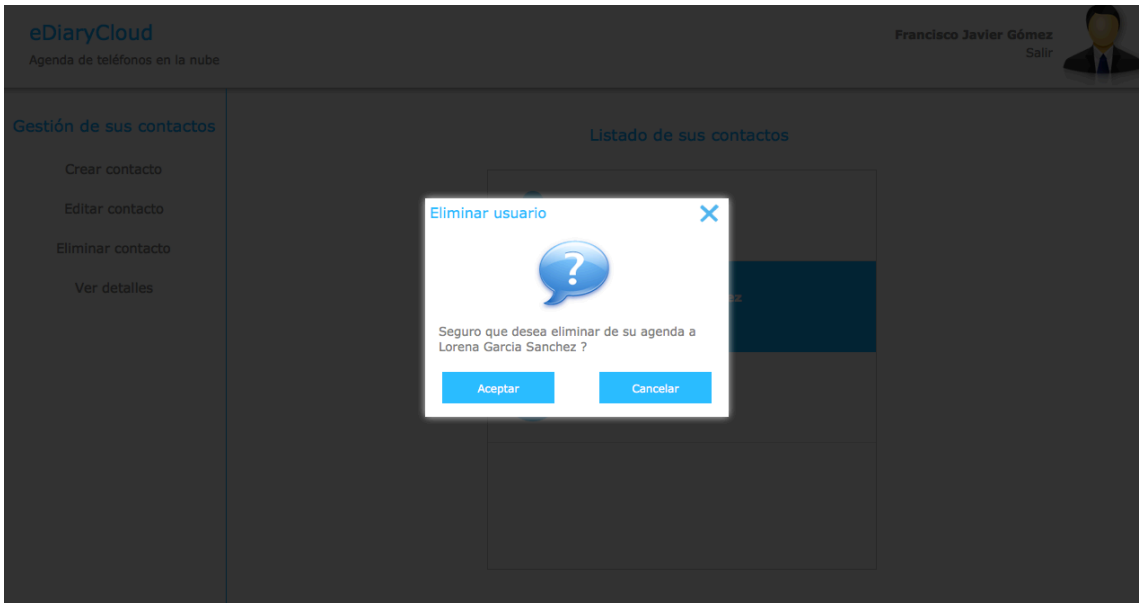

Figura 4.4

4.3.5 Vista ver detalles

Igual que las dos vistas anteriores esta vista solo se activa cuando hay un contacto seleccionado.

Se trata de mostrar la información del contacto con detalle, mostrándose todos los campos contemplados ya que en el listado no aparece toda la información.

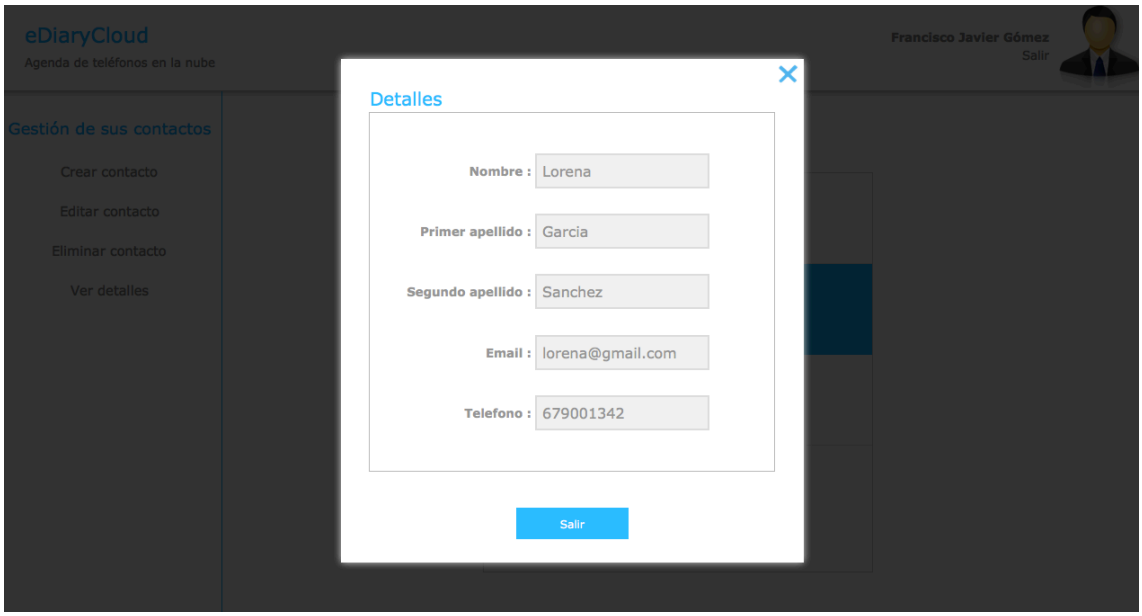

Figura 4.5

Desde esta vista el usuario únicamente puede ver la información disponible para ese contacto y cerrar la ventana para volver a la vista principal.

4.4 Implementación

4.4.1 Arquitectura

Una parte importante del diseño del proyecto es la arquitectura. En este proyecto podemos observar claramente una arquitectura cliente-servidor muy definida gracias a GWT.

Toda aplicación contiene código de presentación, código de procesamiento de base de datos y código de almacenamiento de datos. La arquitectura de las aplicaciones difiere según esta distribuido este código. La arquitectura se basa de tres:

• **Capa de Interfaz.** Esta capa soporta los servicios de presentación, los cuales proporcionan la interfaz necesaria para presentar información y reunir datos. También asegura los servicios de negocio necesarios para ofrecer las capacidades de transacciones requeridas e integrar al usuario con la aplicación para ejecutar un proceso de negocio. El cliente proporciona el contexto de interfaz, que generalmente puede ser un navegador Web, como Internet Explorer, Mozilla Firefox o Google Chrome, los cuales permiten ver los datos remotos a través de una capa de presentación HTML. Esto ayuda a asegurar que los desarrolladores estén libres a la hora de escribir la lógica de negocios sin preocuparse acerca de cómo se muestra en la salida.

• **Capa de Negocio.** Una tarea de negocios es una operación definida por los requerimientos de la aplicación, como pasar los datos del servicio a la base de datos. Las de negocio son políticas que controlan el flujo de las tareas. Los servicios de negocio son el puente entre un usuario y los servicios de datos.

• **Capa de Datos.** La capa de datos se encuentra enlazada con la capa de negocio. El nivel de servicios de datos es responsable de:

- Almacenar los datos.
- Recuperar los datos.
- Mantener los datos.
- Integridad de los datos.

4.4.2 Estructura y código

En cualquier proyecto de GWT los paquetes que cuelgan de 'client' son los que contienen las clases JAVA referentes a la interfaz gráfica y todo lo que cuelga de 'server' son clases referentes a la parte servidora.

Utilizamos como nombre base de paquete 'com.indenova.etsinf.ediarycloud'.

com.indenova.etsinf.ediarycloud.EDiaryCloud.gwt.xml.- El contenido del archivo .gwt.xml especifica una lista precisa de las clases Java y otros recursos que son incluidas en el módulo GWT.

En otras palabras, agrupa todas las propiedades de configuración que el proyecto necesita, entre ellas podemos destacar las siguientes:

- Lista de módulos heredados. Por ejemplo, com.google.gwt.user.User debe estar definido, por ser el núcleo de GWT.
- El nombre de la clase Java que se utilizará como punto de entrada (entry point). Esta es la clase que se instanciará al comienzo, invocando el método EntryPoint.onModuleLoad(), cuando el módulo sea cargado.
- Ubicación de los subpaquetes fuentes a ser compilados y traducidos a JavaScript.
- Ubicación de los subpaquetes públicos a ser compilados y traducidos a Javscript. Aquí debe incluirse el URL completo de la ubicación de estos subpaquetes.
- Además, puede agregar propiedades adicionales para la compilación y traducción de los fuentes en Javascript.

com.indenova.etsinf.ediarycloud.client.- En esta paquete y sus subpaquetes podemos encontrar todos los widgets que hemos creado para la aplicación(Botones, Grids, Labels, Formularios, Ventanas, Imágenes…etc).

com.indenova.etsinf.ediarycloud.client.data.- Contiene los beans de datos que vamos a devolver a la parte cliente de gwt. Son clases Java que deben de implementar la interfaz “IsSerializable”. Ejemplo:

```
public class TelefonoGWT implements IsSerializable {

	private String idTelefono;
	private String numero;

	public TelefonoGWT() {
	}

	public String getIdTelefono() {
		return idTelefono;
	}

	public void setIdTelefono(String idTelefono) {
		this.idTelefono = idTelefono;
	}

	public String getNumero() {
		return numero;
	}

	public void setNumero(String numero) {
		this.numero = numero;
	}
}
```

com.indenova.etsinf.ediarycloud.shared.entities.- Contiene las entities con anotaciones JPA, utilizadas para pasarlas al datastore de App Engine que realizará la persistencia en base de datos. Ejemplo:

```
@Entity
public class Telefono implements Serializable {
	private static final long serialVersionUID = 1L;
	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	@Extension(vendorName = "datanucleus", key = "gae.encoded-pk", value =
"true")
	private String idTelefono;

	@ManyToOne(fetch = FetchType.LAZY)
	private Usuario usuario;

	private String numero;

	public Telefono() {
		super();
	}
```

```
	public Telefono(String numero, Usuario usuario) {
		super();
		this.setNumero(numero);
	}

	public String getIdTelefono() {
		return idTelefono;
	}

	public void setIdTelefono(String idTelefono) {
		this.idTelefono = idTelefono;
	}

	public String getNumero() {
		return numero;
	}

	public void setNumero(String numero) {
		this.numero = numero;
	}

	public Usuario getUsuario() {
		return usuario;
	}

	public void setUsuario(Usuario usuario) {
		this.usuario = usuario;
	}
}
```

com.indenova.etsinf.ediarycloud.shared.services.UsuarioServiceInterface.- Esta interface hace parte de la implementación RPC que hará posible el diálogo entre el cliente y el servidor. Extiende **RemoteService** de GWT y define la lista de los métodos RPC que tendrán una implementación en el servidor.

```
@RemoteServiceRelativePath("UsuarioServiceServlet") → Nombre del servlet
definido en el web.xml que utilizará nuestra aplicación

public interface UsuarioServiceInterface extends RemoteService {

	public UsuarioGWT createUser(UsuarioGWT usuario);

	public List<UsuarioGWT> listUsers();

	public Boolean deleteUser(String idusuario);

}
```

com.indenova.etsinf.ediarycloud.shared.services.UsuarioServiceInterfaceAsync.- Esta es una interface asincrónica del servicio creado en **UsuarioServiceInterface** y es invocada desde el lado del cliente.

```
public interface UsuarioServiceInterfaceAsync {

	void createUser(UsuarioGWT usuario, AsyncCallback<UsuarioGWT> callback);
```

```
        void listUsers(AsyncCallback<List<UsuarioGWT>> callback);

        void deleteUser(String idusuario, AsyncCallback<Boolean> callback);

}
```

com.upv.etsinf.ediarycloud.server.persistence.UsuarioServiceInterfaceImpl.- Esta clase define el servicio RPC en si. Extiende de **RemoteServiceServlet** e implementa la interface asociada que define los servicios. Obsérvese que la implementación del servicio no se hace sobre la versión asincrónica de la interface. Ejemplo de creación de un usuario.

```
public class UsuarioServiceInterfaceImpl extends RemoteServiceServlet
implements UsuarioServiceInterface {

        EntityManagerFactory emf = null;

        public UsuarioServiceInterfaceImpl() {
                emf = Persistence.createEntityManagerFactory("transactions-
optional");
        }

        @Override
        public UsuarioGWT createUser(UsuarioGWT usuarioGWT) {
                EntityManager entityManager = null;
                Usuario usuario = new Usuario();
                usuario.setNombrePrimero(usuarioGWT.getNombrePrimero());
                usuario.setNombreSegundo(usuarioGWT.getNombreSegundo());
                usuario.setApellidoPaterno(usuarioGWT.getApellidoPaterno());
                usuario.setApellidoMaterno(usuarioGWT.getApellidoMaterno());
                usuario.setEmail(new Email(usuarioGWT.getEmial()));
                for (int i = 0; i < usuarioGWT.getTelefonos().size(); i++) {
                        Telefono tel = new Telefono();

        tel.setNumero(usuarioGWT.getTelefonos().get(i).getNumero());
                        usuario.getTelefonos().add(tel);

                }

                entityManager = emf.createEntityManager();
                entityManager.getTransaction().begin();
                entityManager.persist(usuario);
                entityManager.flush();
                entityManager.getTransaction().commit();
                usuarioGWT.setidUsuario(usuario.getIdUsuario());
                return usuarioGWT;

        }
```

En este ejemplo podemos ver que instanciamos un **EntityManagerFactory** para luego crear un **EntityManager**, es decir, hacemos uso de las herramientas de JPA sin preocuparnos de cómo funciona la base de datos de APP Engine. Este código en servidor que persiste un usuario es el mismo que podría tener otra aplicación con otra base de datos distinta.

La única diferencia es que debido al uso de gwt nos vemos obligados a convertir el objeto “UsuarioGWT” que nos llega de la parte cliente, en “Usuario”, para poder tener una entitie JPA a persistir.

Carpeta META-INF.- Contiene el fichero **persistence.xml.** Aquí se define:

- Nombre de la unidad de persistencia que luego se usa para instanciar el **EntityManagerFactory**.
- Proveedor de JPA para App Engine.
- Entities usadas para persistir en base de datos.
- Propiedades propias de a base de datos de App Engine, como por ejemplo políticas de escritura-lectura y URL de conexión a la base de datos.

Ejemplo de mi persistente.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
        http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd" version="1.0">

    <persistence-unit name="transactions-optional">
        <provider>org.datanucleus.api.jpa.PersistenceProviderImpl</provider>
        <class>com.upv.etsinf.ediarycloud.shared.entities.Usuario</class>
           <class>com.upv.etsinf.ediarycloud.shared.entities.Telefono</class>
           <exclude-unlisted-classes>true</exclude-unlisted-classes>
        <properties>
            <property name="datanucleus.NontransactionalRead" value="true"/>
            <property name="datanucleus.NontransactionalWrite" value="true"/>
            <property name="datanucleus.ConnectionURL" value="appengine"/>
        </properties>
    </persistence-unit>
</persistence>
```

Carpeta WEB-INF.- Contiene los recursos estáticos de la aplicación como los CSS, imágenes y el .html que se carga al inicio.

Mostramos una imagen ilustrativa de la estructura del proyecto

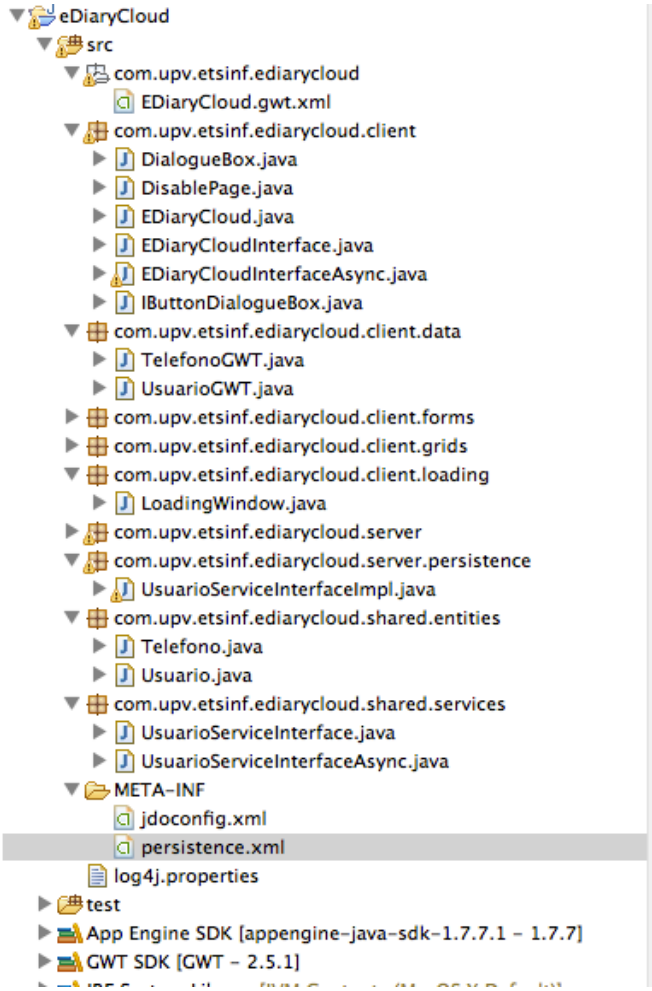

Figura 4.6

5. Metodología propuesta

5.1 Instalación del plugin de App Engine

Google ofrece un plugin que proporciona todo lo necesario para crear desarrollos en Google App Engine con la tecnología GWT .

Para instalar el plugin debemos acceder al gestor de actualizaciones de **Eclipse Helios** y añadir http://dl.google.com/eclipse/plugin/archive/3.2.4/3.6.

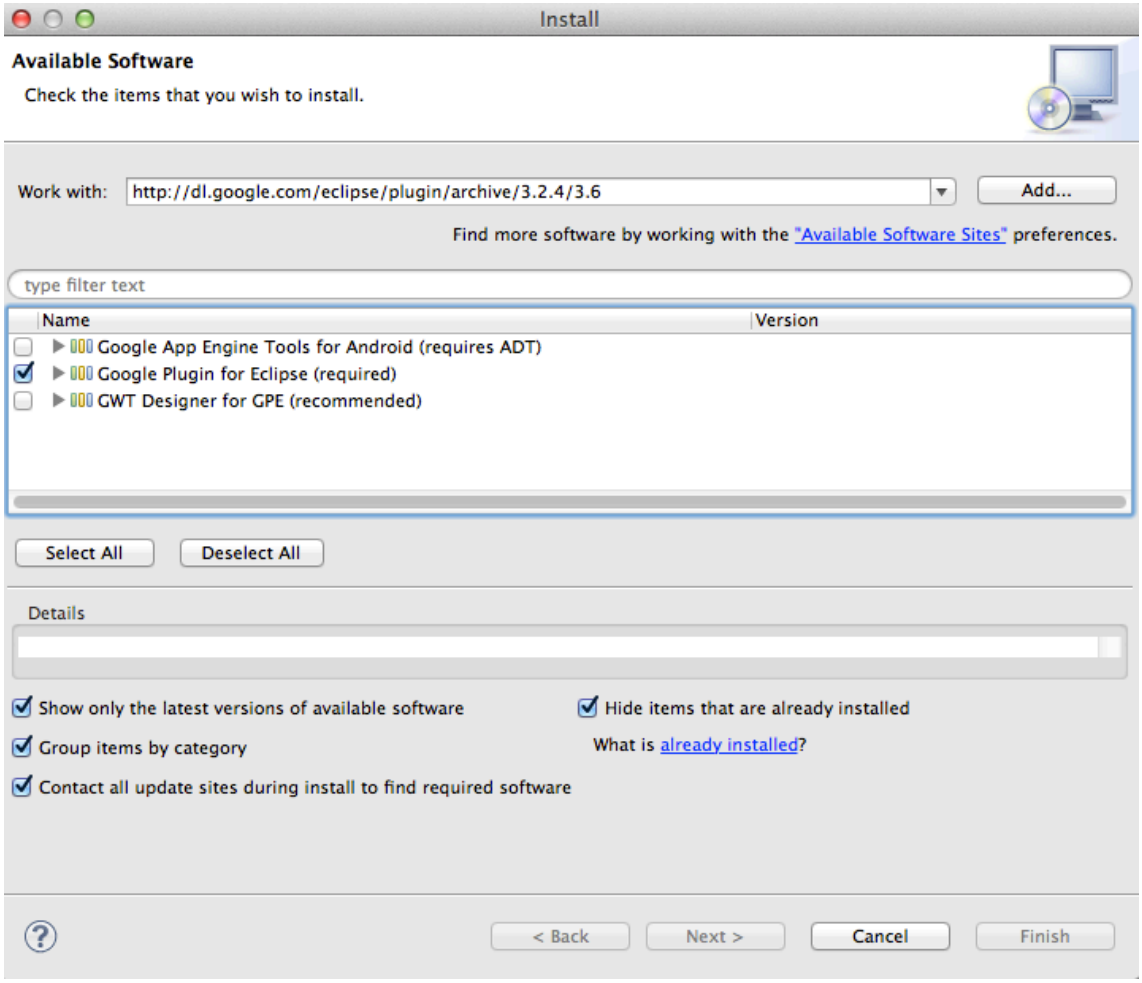

Figura 5.1

Seleccionamos “Google Plugin for Eclipse”, pulsamos en siguiente y al finalizar el asistente ya tendremos el plugin instalado.

5.2 Registrarse en Google App Engine

Para implementar la aplicación en la nube de Google se necesita una cuenta de App Engine. Para obtener este tipo de cuenta es necesario tener una cuenta de correo electrónico de Google.

Abrir la URL http://appengine.google.com/ e identificarse con la información de tu cuenta Google. Se verifica la cuenta a través de un número de teléfono válido. Después de proporcionar el número de teléfono, Google envía un mensaje de texto con un código de verificación. A continuación, se escribe el código de verificación en la casilla de verificación en línea.

Finalmente ya estamos registrados en App Engine, a partir de ahora ya podemos subir nuestra aplicación a los servidores de Google y disponemos de una consola de administración que nos permite gestionar nuestras aplicaciones.

Este es un ejemplo de nuestra cuenta creada.

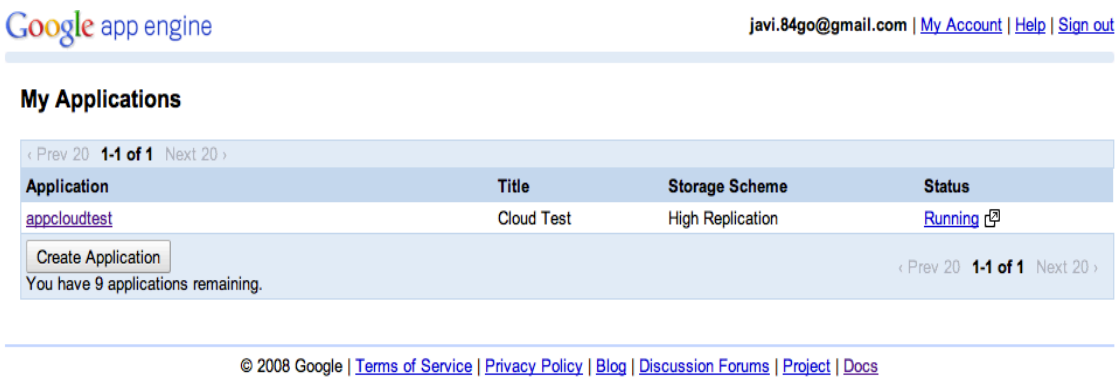

Figura 5.2

5.3 Crear la aplicación en App Engine

El proceso de creación de una aplicación en la App Engine Google supone la creación de una aplicación en el sitio web de Google App Engine. Después se puede crear localmente una aplicación web y subir esta aplicación en el sitio creado en App Engine.

Para crear una aplicación pulse el botón "Crear una aplicación" y seleccione un nombre de aplicación. Se tiene que elegir uno que todavía está disponible.

Básicamente este paso es como comprar un dominio y hosting de nuestra aplicación cuando creamos una aplicación Web tradicional y queremos subirla al servidor.

Hay que recordar que el nombre que pongamos al crear la aplicación será el nombre de dominio que aparecerá en la url de acceso.

Ejemplo:

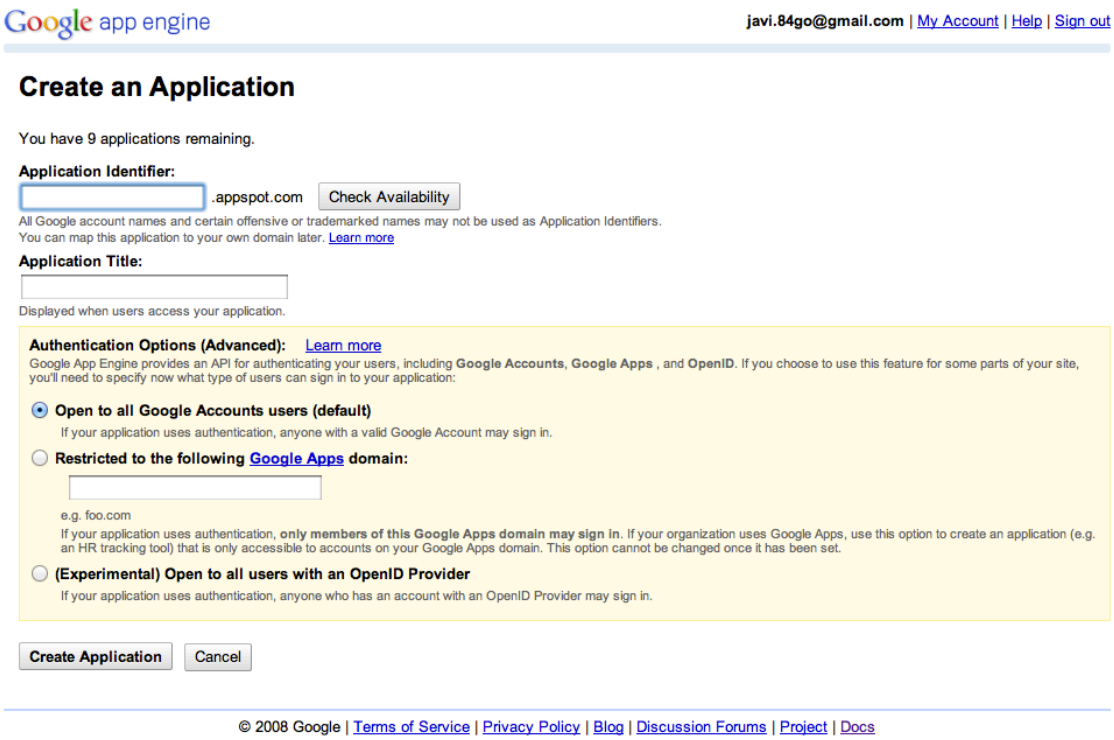

Figura 5.3

5.4 Crear el proyecto en Eclipse

Ir a Archivo->Nuevo->Google->Web Aplication Proyect

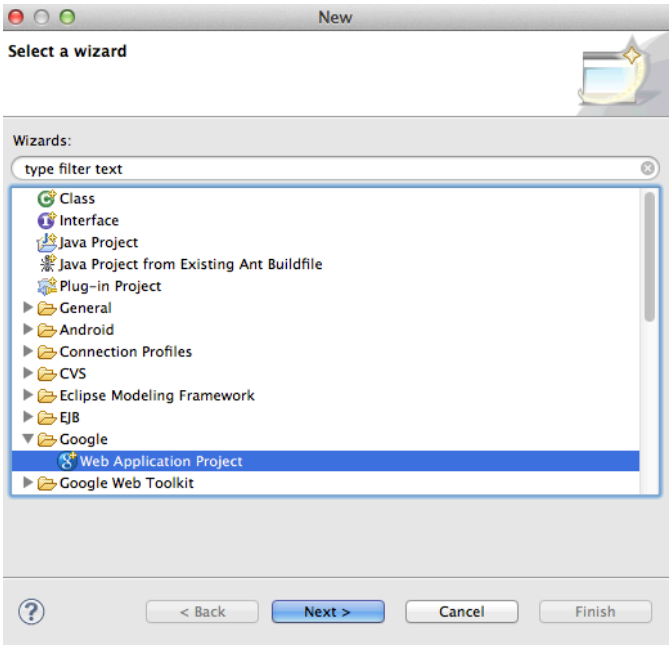

Figura 5.4

En el siguiente paso hay que escribir un nombre para el proyecto, un paquete base y en el apartado Google SDKs marcar “Use Google Web Toolkit” y “Use Google App Engine”. Esto nos generará un proyecto preparado para desarrollar nuestra aplicación en App Engine con la tecnología GWT.

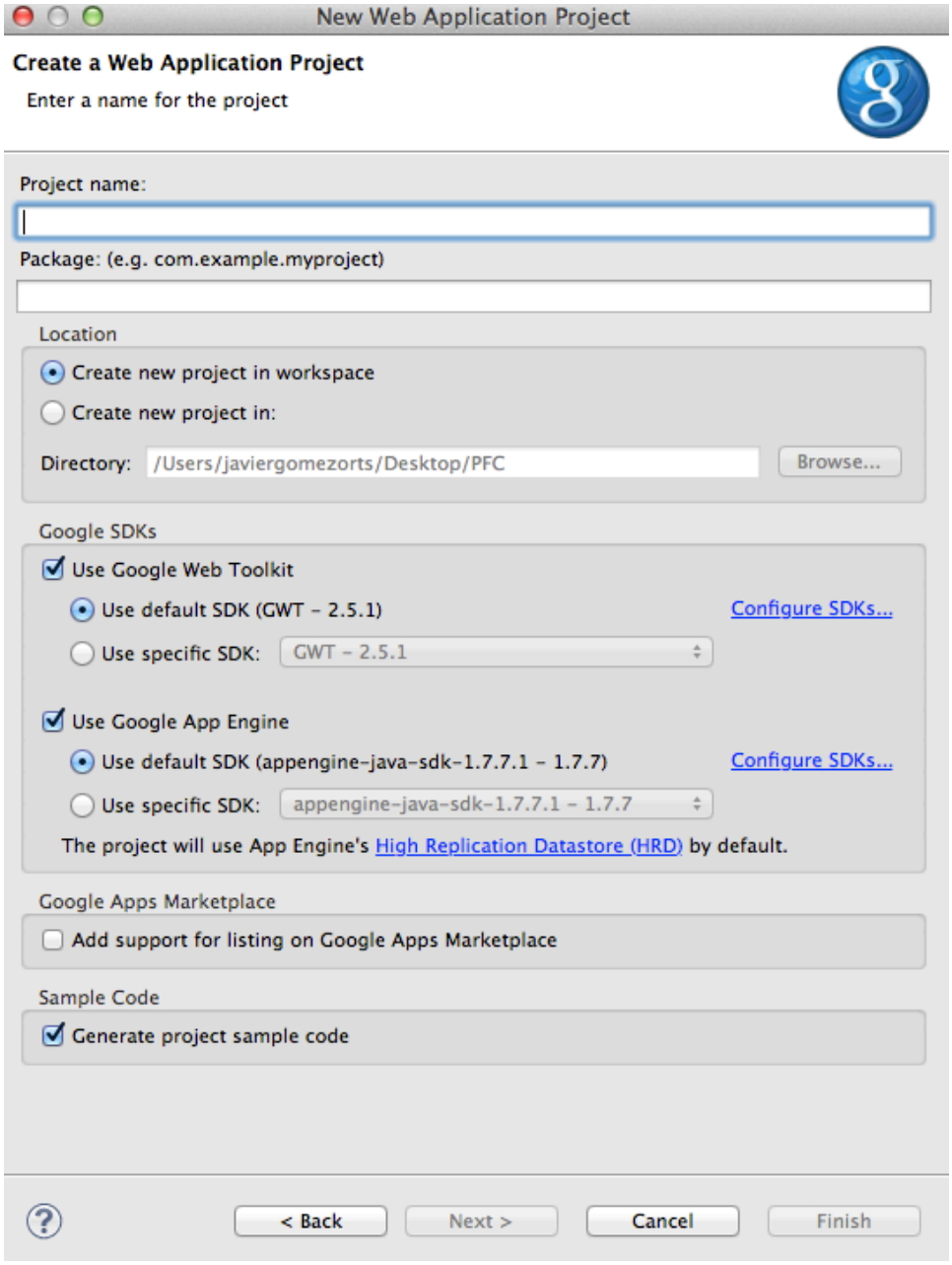

Figura 5.5

Finalmente ya tendremos creado nuestro proyecto para comenzar con el desarrollo.

5.5 Ejecutar la aplicación en local

El plugin nos permite ejecutar la aplicación en local simulando un entorno real en la nube de Google.

Debemos seleccionar nuestro proyecto, abrir el panel de propiedades y seleccionar “Run As->Web Application”.

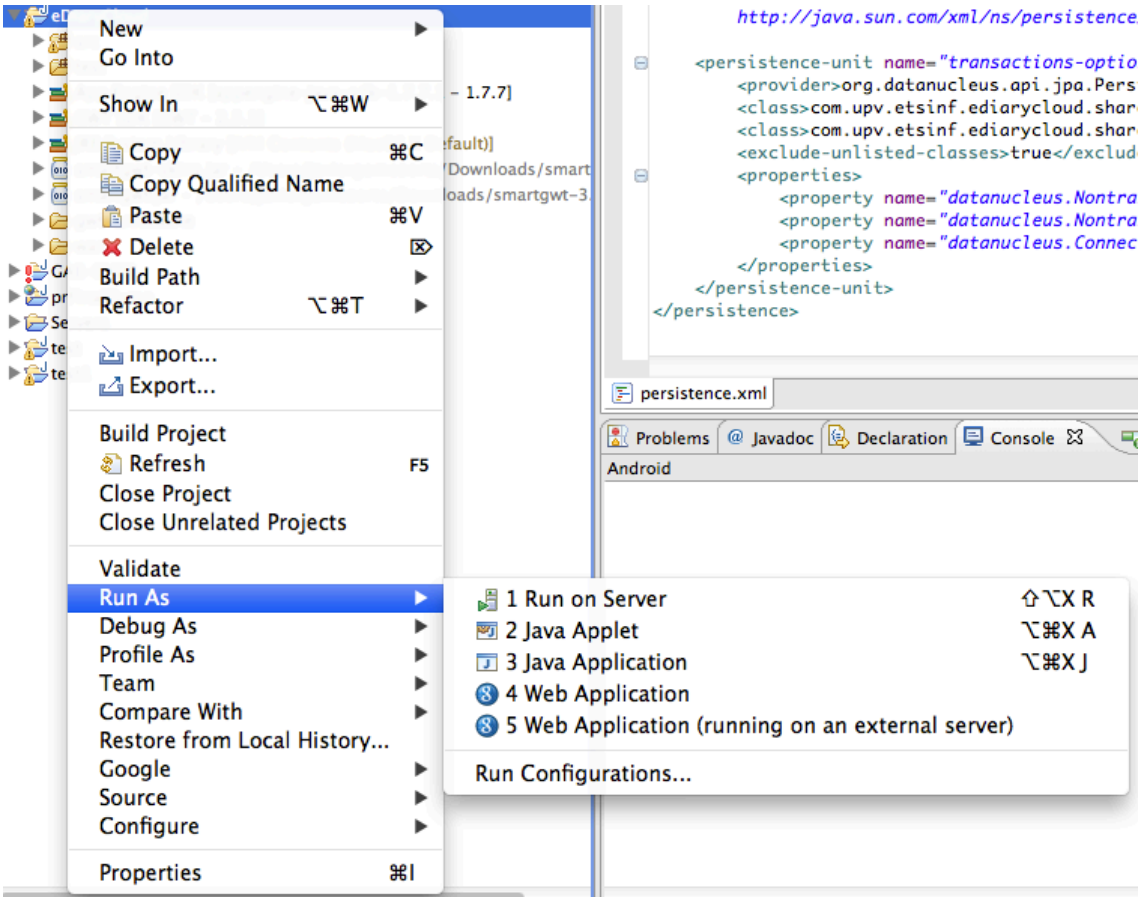

Figura 5.6

Finalmente se abrirá el navegador por defecto con la url http://localhost:8888/EDiaryCloud.html.

También podemos acceder a la consola de administración del datastore mediante la siguiente url http://localhost:8888/_ah/admin.

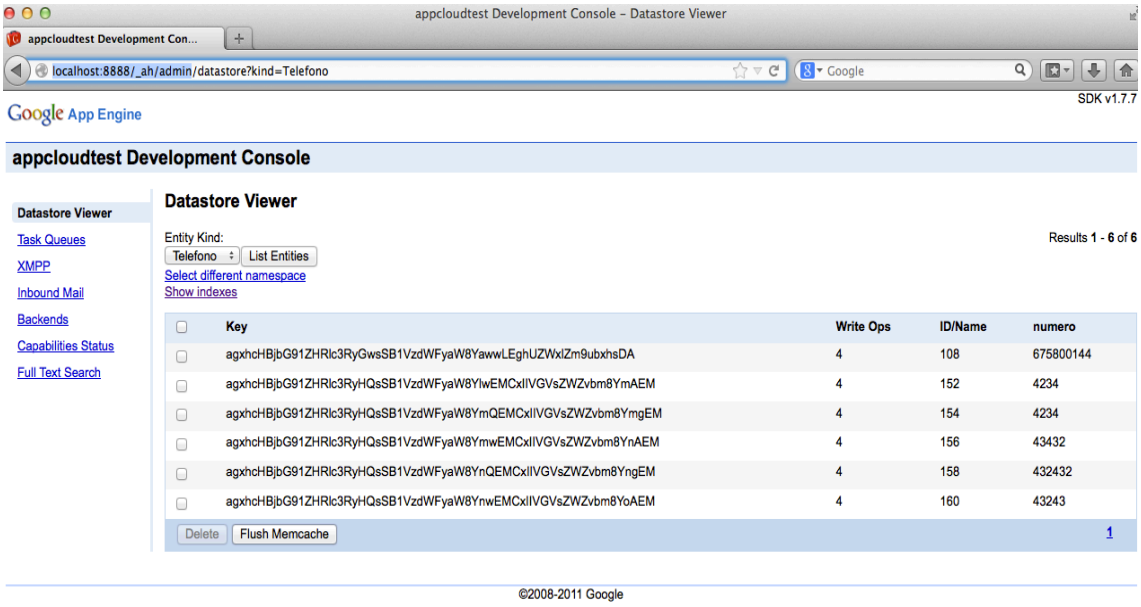

Figura 5.7

5.6 Desplegar la aplicación en App Engine

Para desplegar la aplicación en los servidores de Google debemos abrir el panel de propiedades de nuestro proyecto y seleccionar Google->Deploy to App Engine. Este proceso tarda ya que se ejecuta el compilador de GWT, el cual traduce todo el código a javascript y después se comienza a subir todos los archivos.

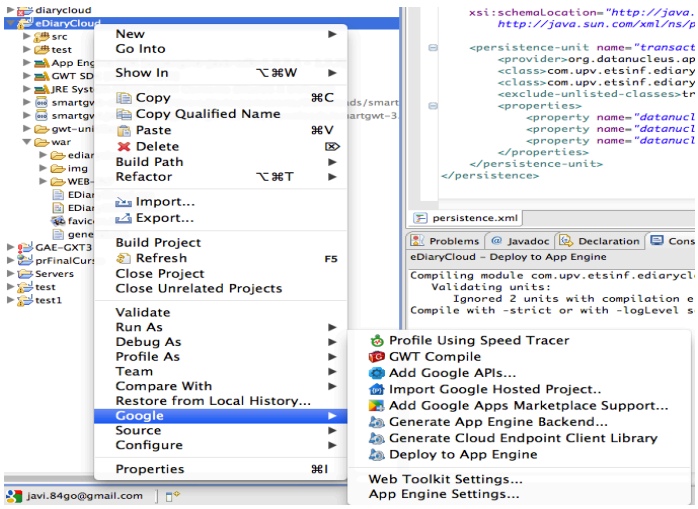

Figura 5.7

5.7 Verificar que funciona

Se van a describir unas secuencias de pasos para testear la aplicación.

Cuando nos referimos con un numero de paso mas una “a”(por ejemplo 2a), estamos indicando que es un segundo paso alternativo y por lo tanto todos los pasos que continúen por esa rama acabarán en “a”.

5.7.1 Caso de prueba listado de contactos

Pasos para verificar que se están listando los contactos existentes en la base de datos de App Engine.

Precondinciones: Ninguna, en un caso completo el usuario debería haberse logado previamente en el sistema.

Secuencia de pasos:

1. Cargar aplicación accediendo a la URL http://ediarycloud.appspot.com.
2. Comprobar que el listado de contactos coincide con el numero de usuarios almacenados en base de datos.
   Podemos acceder al panel de administración de base de datos desde la siguiente url http://ediarycloud.appspot.com/, se nos muestra un listado con las aplicaciones dadas de alta en App Engine, debemos pinchar en nuestra aplicación y luego en la sección “**Datastore Admin**”.

2a.- Si no hay contactos el sistema debe mostrar un mensaje diciendo “No hay contactos en la agenda”.

5.7.2 Caso de prueba crear contacto

Precondiciones: Ninguna.

Secuencia de pasos:

1. Pulsar en la opción “Crear contacto” de la barra de menú lateral.

2. Rellenar formulario con los siguientes datos (Nombre->nombretest, Primer apellido-> apellido test, Segundo apellido-> apellido2test, Emial-> emailtest@gmail.com, Teléfono-> 678012433) y pulsar en “Guardar”.

2a. Rellenar formulario con los siguientes datos (Nombre->"dejar vacío", Primer apellido-> apellido test, Segundo apellido-> apellido2test, Emial-> emailtestgmail, Teléfono-> aaa67800) y pulsar en "Guardar".

3a. El sistema notifica al usuario los campos incorrectos y suspende la creación del contacto.

3. El sistema realiza el proceso de persistencia.

4. Se notifica al usuario que el contacto se ha creado correctamente.

5. Pulsar en "Aceptar".

6. El listado de contactos se actualiza.

5.7.3 Caso de prueba editar contacto

Precondiciones: Ninguna.

Secuencia de pasos:

1. Seleccionar un contacto y pulsar en la opción "Editar contacto" de la barra de menú lateral.
2. Modificar formulario con lo siguientes datos (Nombre->nombremod, Primer apellido-> apellidomod, Segundo apellido-> apellido2mod, Emial-> emailtestmod@gmail.com, Teléfono-> 679011222) y pulsar en "Guardar".
3. El sistema el proceso de actualización.
4. Se notifica al usuario de que el contacto se ha modificado correctamente.
5. Pulsar en "Aceptar".
6. El listado de contactos se actualiza.

5.7.4 Caso de prueba eliminar contacto

Precondiciones: Ninguna.

Secuencia de pasos:

1. Seleccionar un contacto y pulsar en la opción "Eliminar contacto" de la barra lateral de menú.
2. Sistema pregunta al usuario si realmente desea eliminar el contacto.

2a. Se Pulsa en “Cancelar” y el sistema suspende la eliminación del contacto.

3. Se pulsa en “Aceptar”.
4. El sistema elimina el contacto de la base de datos.
5. Se actualiza el listado.

5.7.5 Caso de prueba ver detalles

Precondiciones: Ninguna.

Secuencia de pasos:

1. Seleccionar un contacto y pulsar en la opción “Ver detalles” de la barra lateral de menú.
2. Sistema muestra formulario con los datos del contacto.
3. Sistema no permite editar los datos.
4. Usuario pulsa en “Salir” y vuelve al listado.

6. Conclusiones

En esta memoria, primero se han presentado las tecnologías implicadas, en la implementación de una aplicación en la nube, para después exponer nuestra aplicación de ejemplo y una metodología. Se ha explicado el concepto de nube y se han detallado los servicios que ofrece esta tecnología. La aplicación de ejemplo muestra con lujo de detalle los aspectos clave para diseñar e implementar una aplicación Web en la nube, utilizando las herramientas de Google.

En la implementación de la aplicación nos hemos encontrado con algunos problemas, al intentar implementar la capa de persistencia con JPA, debido a las limitaciones de esta tecnología.

Otro problema que nos ha surgido, es al utilizar una entidad y después hacer una consulta a la base de datos para obtener todas los registros. Este problema es el llamado “Eventually inconsistence” y se produce cuando se utiliza “Hight Replication” como sistema de almacenamiento del datastore. La solución a este problema con JPA no es fácil ya que hay que usar consultas “ancestro”.

Por estos motivos yo no recomiendo usar JPA para implementar la capa de persistencia en App Engine, es cierto que para una aplicación sencilla se pueden solventar, pero si la aplicación es mas compleja considero que hay alternativas mejores.

Si que me parece interesante implementar la aplicación en JAVA, ya que dispone de un Framework muy potente como es GWT, que facilita mucho el desarrollo de la parte cliente, dando un aspecto profesional y aportando una mejor experiencia al usuario. Con este Framework se pueden crear aplicaciones complejas ya que esta pensado para el desarrollo de aplicaciones Web enriquecidas, es decir, semejantes a una aplicación de escritorio convencional.

En aspectos generales ha sido satisfactorio el desarrollo de la aplicación en la nube con JAVA y he echado de menos la falta de tiempo para poder crear una aplicación mas completa.

7. Bibliografía

Página de documentación técnica del mundo de la ingeniería y la tecnología.

http://ieeexplore.ieee.org/Xplore/guesthome.jsp.

Documentación oficial de Google sobre App Engine

https://developers.google.com/appengine/docs/java/datastore/overview?hl=es

https://developers.google.com/appengine/docs/whatisgoogleappengine?hl=es

Página con información relativa a Google Web Toolkit (GWT)

http://www.gwtproject.org/

Anexo

Lista de figuras