Programación en la Nube

J.A. Medina

Ciencias de la Computación

Universidad de Alcalá

Introducción

Objetivo

- Conocer y utilizar la arquitectura de la nube
- Ser capaces de diseñar, implementar y publicar una aplicación haciendo uso de los servicios más importantes ofrecidos por:
  - Google Cloud - App Engine
  - Amazon Web Service
  - Azure

Recursos en la Web

- Aula virtual de Uah

  Código de ejemplo, transparencias empleadas en clase, ejercicios, planificación,…

- App Engine

  SDK, documentación, complementos, ejemplos, …

- Amazon Web Service AWS (AWS Cost Calculator)
- Windows Azure

Bibliografía

- **Chandrasekaran , Essentials of Cloud Computing, CRC Press , 2015.**
- **Roche & Douglas, “Beginning Java Google App Engine”, Apress, 2009.**
- Dewsbury, “Google Web Toolkit Applications”, Prentice-Hall, 2008.
- Chen-Becker, Danciu & Weir, “The definite Guide to Lift”, Apress, 2009.
- **Collier & Shahan, “Microsoft Azure Essentials: Fundamentals of Azure, Second Edition”, Microsoft Press, 2016 https://mva.microsoft.com/ebooks#9780735697225.**
- **Amazon Web Services, Getting Started with AWS. 2016 http://docs.aws.amazon.com/gettingstarted/latest/awsgsg-intro/**
- **AWS Whitepapers Architecting for the AWS Cloud: Best Practices, 2018 https://aws.amazon.com/whitepapers/**

En la nubes

- **Cloud computing** está basado en la computación en Internet, donde los recursos se comparten, el software y la información son proporcionados a computadoras y otros dispositivos bajo demanda.
- Es la culminación de numerosos intentos de computación a gran escala con acceso a una cantidad ilimitada de recursos.
  - on-demand computing, ubiquitous computing, autonomic computing, platform computing, edge computing, elastic computing, utility computing, **grid computing**, …

En la nubes

- **on-demand computing**,
  - Es un modelo empresarial cada vez más popular
  - Ponen a disposición del usuario los recursos de computación como sea necesario
  - Los recursos se pueden mantener en la empresa del usuario, o puestos a disposición por un proveedor de servicios
- **ubiquitous computing,**
  - La integración de la informática en el entorno de la persona, de forma que los ordenadores no se perciban como objetos diferenciados
- **autonomic computing,**
  - nuevo paradigma, cambia de un paradigma centrado en los equipos, a uno centrado en los datos.
- **platform computing,**
  - soluciones y servicios de gestión de sistemas de baja latencia y alto rendimiento ej: IBM Platform Computing
- **edge computing,**
  - la totalidad o la mayor parte de los datos en la red son impulsados lejos de equipos físicos, por lo que utiliza principalmente la red para almacenar su información
- **elastic computing,**
  - La habilidad de aumentar o disminuir dinámicamente los recursos de procesamiento, memoria y almacenamiento para satisfacer las demandas
- **utility computing,**
  - suministro de recursos computacionales, como puede ser el procesamiento y almacenamiento, como un servicio medido similar a las utilidades públicas tradicionales (como la electricidad, el agua, el gas natural o el teléfono).
- **grid computing,**
  - utilizar de forma coordinada todo tipo de recursos (entre ellos cómputo, almacenamiento y aplicaciones específicas) que no están sujetos a un control centralizado

Algo está cambiando…

- **Crecimiento exponencial** en las aplicaciones: biomedicina, exploración espacio, business analytics, web 2.0 social networking: YouTube, Facebook,…
- Generación de **contenidos escalable**: e-science and e-business data
- Gran **ratio de consumo de contenido digital**: Apple iPhone, iPad, Amazon Kindle,…
- Crecimiento exponencial en las **capacidades de computo**: multi-core, storage, bandwidth, virtual machines (virtualization)
- **Ciclos muy cortos de obsolescencia**: Windows Vista→ Windows 7; Java versions; Phython
- **Nuevas arquitecturas**: web services, modelos de persistencia, sistemas de ficheros distribuidos/repositorios (Google, Hadoop), multi-core, wireless,…
- No se puede manejar situaciones complejas con la infraestructura tradicional

Problemas de Existentes.

Cuando Empresa **contrata un desarrollo a medida** necesita:

- Un servidor que se encuentra físicamente dentro del propio edificio de la organización.
- Desplazamiento actualización de versiones
- Servidor dedicado más otro para datos (2 servidores)
- Un responsable de gestionar sus sistemas, que se encargara de realizar las copias de seguridad, de que los equipos funcionen correctamente, de tener repuestos por si alguno de los sistemas fallase, etc.
- Este tipo de aplicaciones son utilizadas por empleados que posiblemente no se encuentren en el mismo edificio en el que esté el servidor de la aplicación, por lo que se debe mantener una conexión a Internet que funcione constantemente, con un ancho de banda adecuado, y mantener unas medidas de seguridad,etc.
- Y si la aplicación se hace más grande, la base de datos crece mucho, o se empiezan a almacenar muchos ficheros de gran tamaño posiblemente el servidor deba cambiarse

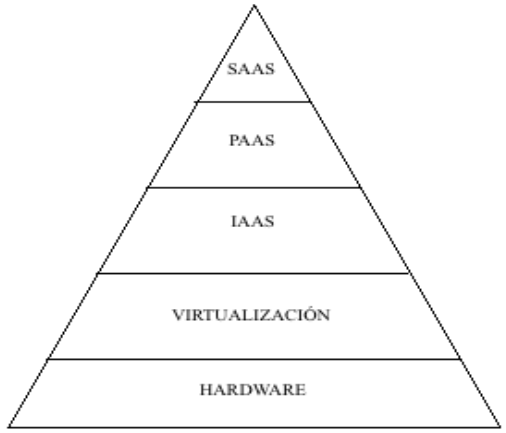
Cloud Computing for THE IT CROWD

- Revisar coste del modelo frente a utilidad: CPU/hour, GB/day etc.
- Incluir costes de mantenimiento, formación,…
- Nubes diferentes para distintas aplicaciones
- Desarrollar prototipos

…

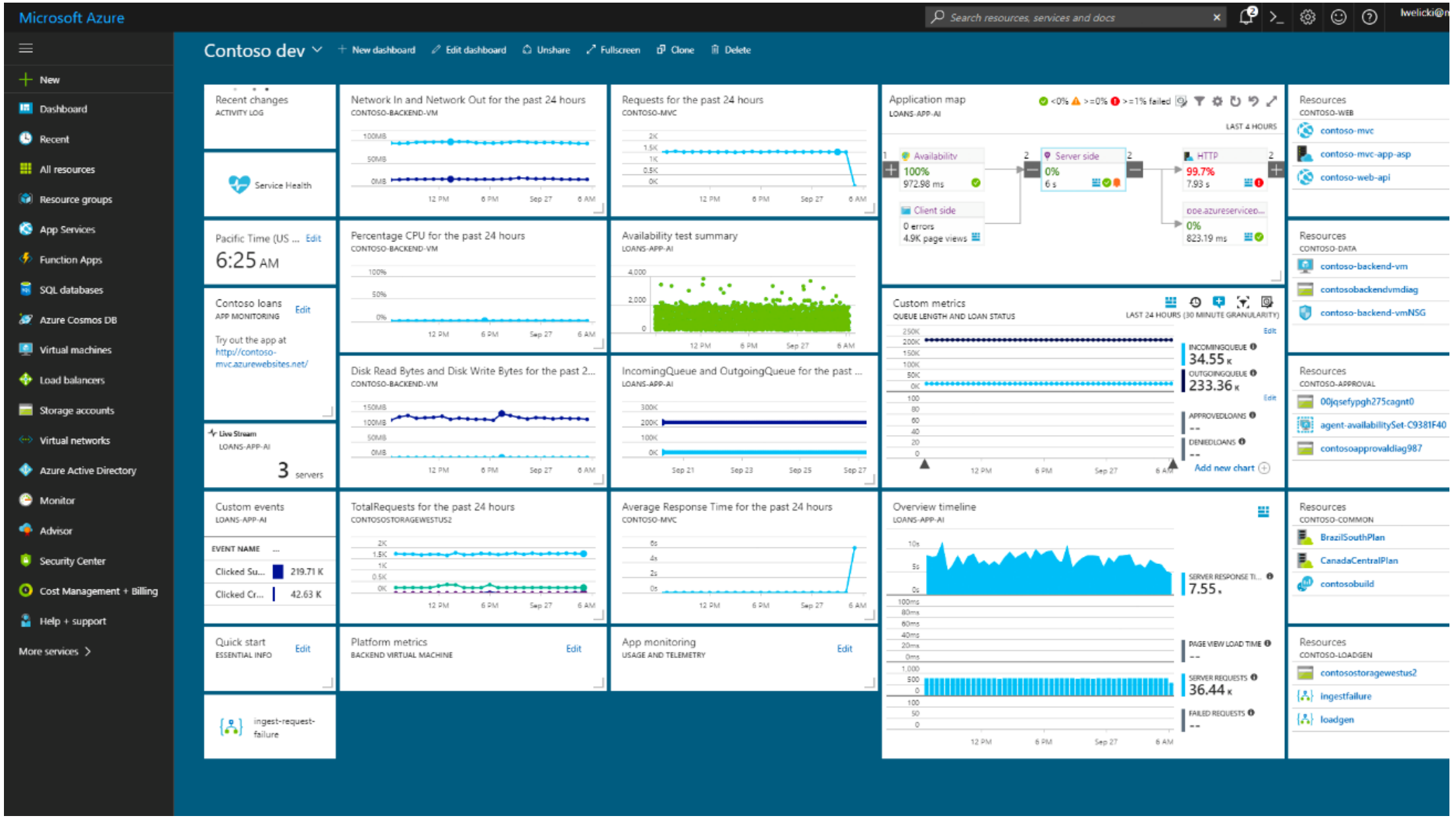
Solución: Cloud Computing

- Requerimientos y modelos típicos:
  - software (SaaS),
  - platform (PaaS),
  - infrastructure (IaaS),

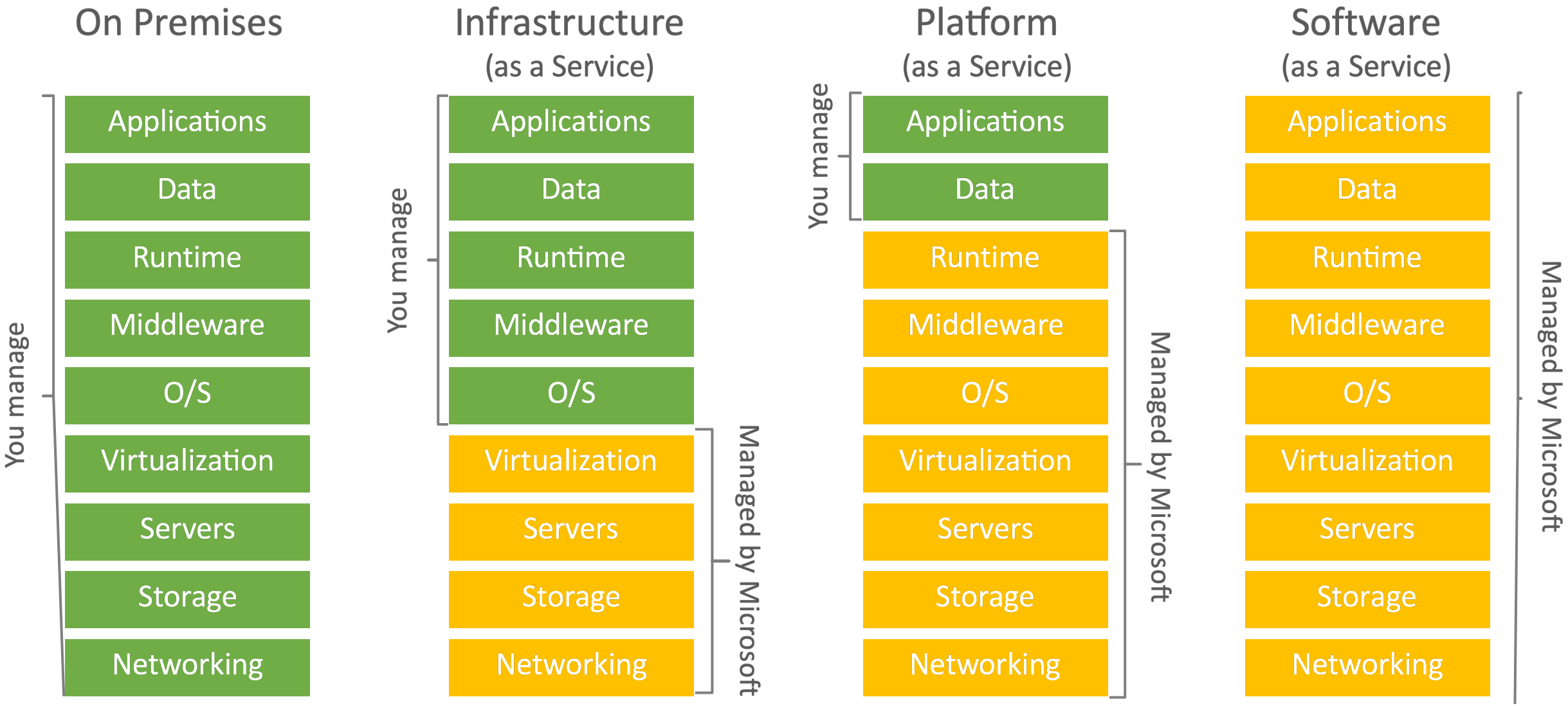


- Services-based application programming interface (API)
- Un entorno cloud provee uno o más de los requerimientos
- Suele facturarse en base al consumo
- Pueden ser públicas o privadas

Solución: Cloud Computing



Modelos de nube

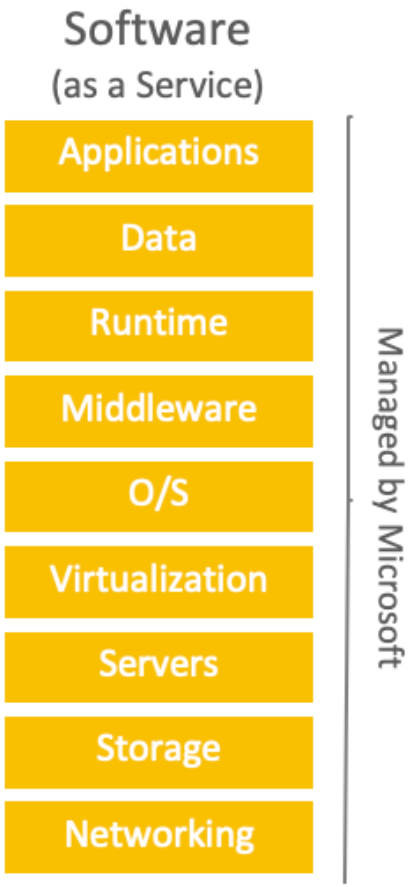


¿Qué es SaaS?

- concepto de Software como Servicio (SaaS, Software as a Service)
- cualquier servicio cloud en el que los consumidores puedan acceder a aplicaciones de software a través de internet
- se conoce también **a veces como "software a demanda"**
- Todo el desarrollo, mantenimiento, actualizaciones, copias de seguridad es responsabilidad del proveedor.
- Alquilar el software en lugar que comprarlo
- Ej: Docs, Salesforce, Dropbox, Gmail…

¿Qué es SaaS?

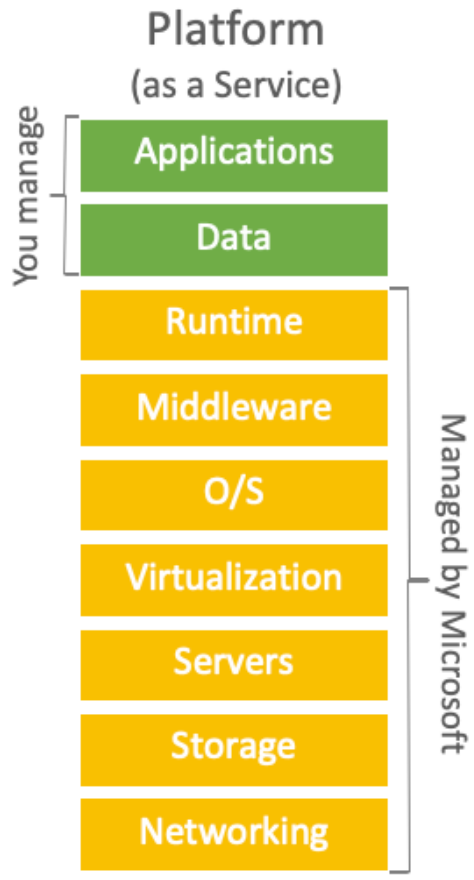
- Ventajas del modelo SaaS para empresas como para particulares:
  - No tiene costes adicionales de hardware
  - No tiene costes de alta
  - Se paga sólo por lo que se utiliza
  - El uso del servicio es escalable
  - Las actualizaciones son automáticas
  - Compatibilidad entre dispositivos
  - Accesible desde cualquier lugar
  - Las aplicaciones pueden personalizarse y asociarse a la imagen de marca del proveedor



¿Qué es PaaS?

- Concepto de Plataforma como Servicio (PaaS, Platform as a Service)
- proporciona una plataforma y un entorno que permiten a los desarrolladores crear aplicaciones y servicios que funcionen a través de internet
- única preocupación es la construcción de nuestra aplicación
- Los servicios PaaS se aloja en la nube y se accede a través de un navegador web
- funcionalidades preconfiguradas a las que los clientes puedan suscribirse
- Ej: Google App Engine, Azure, Amazon Web Service (AWS)

¿Qué es PaaS? (y1)



- Funcionalidades que puede incluir son:
  - Sistema operativo
  - Entorno de scripting de servidor
  - Sistema de gestión de base de datos
  - Software de servidor
  - Soporte técnico
  - Almacenamiento
  - Acceso a la red
  - Herramientas de diseño y desarrollo
  - Hosting
- Ventajas que aporta el modelo PaaS
  - No necesitan invertir en infraestructura física
  - Hace posible que incluso usuarios "no expertos" puedan realizar desarrollos
  - Flexibilidad
  - Adaptabilidad
  - Permite la colaboración entre equipos situados en varios lugares distintos
  - Seguridad

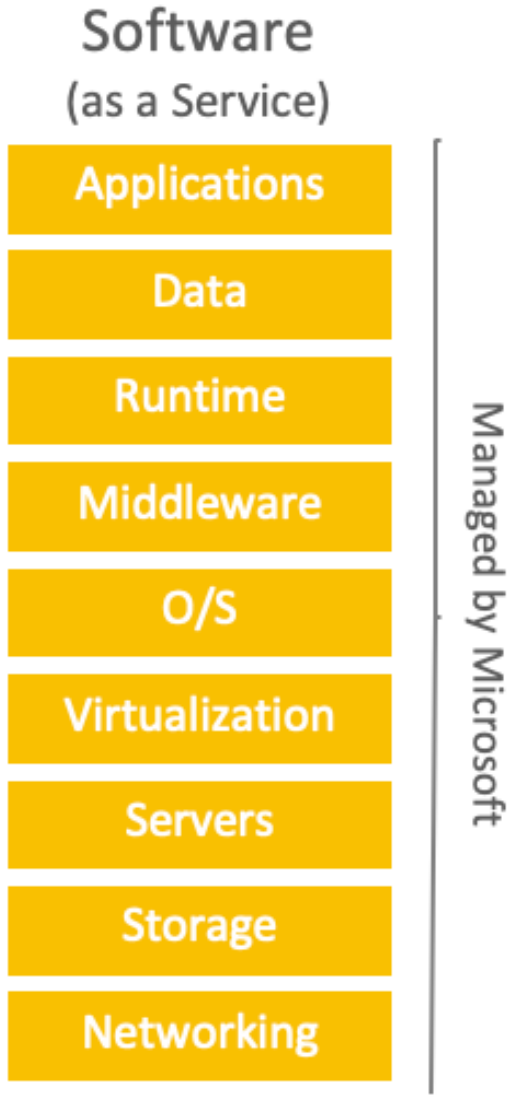
¿Qué es IaaS?

- concepto de Infraestructura como Servicio (IaaS, *Infrastructure as a Service)*
- proporciona acceso a recursos informáticos situados en un entorno virtualizado, la"nube" (cloud)
- aspectos como el espacio en servidores virtuales, conexiones de red, ancho de banda, direcciones IP y balanceadores de carga
- nosotros nos encargamos de escalar nuestras aplicaciones según nuestras necesidades
- El cliente obtiene acceso a los componentes virtualizados para construir con ellos su propia plataforma informática.
- Ej: Amazon Web Service (AWS)- EC2, Azure

¿Qué es IaaS?

- Ventajas
  - Escalabilidad
  - Sin necesidad de invertir en hardware
  - Tarificación similar suministros públicos como luz o gas
  - Independencia de la localidad
  - Seguridad física en los centros de datos
  - No hay puntos únicos de fallo

https://docs.microsoft.com/es-es/azure/storage/blobs/storage-blobs-introduction



¿Nubes privadas, públicas o hibridas?

- nube privada están destinada a un uso exclusivo por parte de la empresa
  - requiere grandes medidas de seguridad tanto de los datos como de la plataforma en la que se almacenan
  - servicio de acceso y disponibilidad muy alto
  - Puede ser utilizado de forma interna (cloud privada interna) o por proveedores (cloud privada externa)
  - Los costes tanto de inversión como de mantenimiento suelen ser más altos que de otras nubes
- nube pública el servicio pertenece a un tercer proveedor y no, a la empresa
  - su uso no solo reside en la propia compañía sino también el suministrador del servicio cloud
  - infraestructura multi-uso, (diferentes usuarios o empresas)
  - forma gratuita o de pago
- Nube hibrida combinan soluciones privadas y públicas

Public Cloud vs. Private Cloud

Ventajas de las privadas:

- Seguridad y privacidad de los datos
- Lock-in del vendedor
- Altos requerimientos computacionales
- Reducción de costes al compartir la infraestructura entre los distintos proyectos de la empresa

Data center

- http://www.aunclicdelastic.com/redundancia-seguridad-y-disponibilidad-claves-del-exito-de-un-datacenter-ii/

- http://www.datacenterdynamics.es/focus/archive/2013/12/microsoft-ampl%C3%ADa-su-data-center-de-dubl%C3%ADn

Ejemplos de Cloud Computing

- http://hacking-etico.com/2013/12/26/wpa-cloud-computing/
- http://www.fayerwayer.com/2011/01/logran-vulnerar-el-cifrado-wpa-psk-en-minutos-gracias-al-cloud-computing/
- http://muycloud.com/2014/03/28/windows-azure-diabetes/

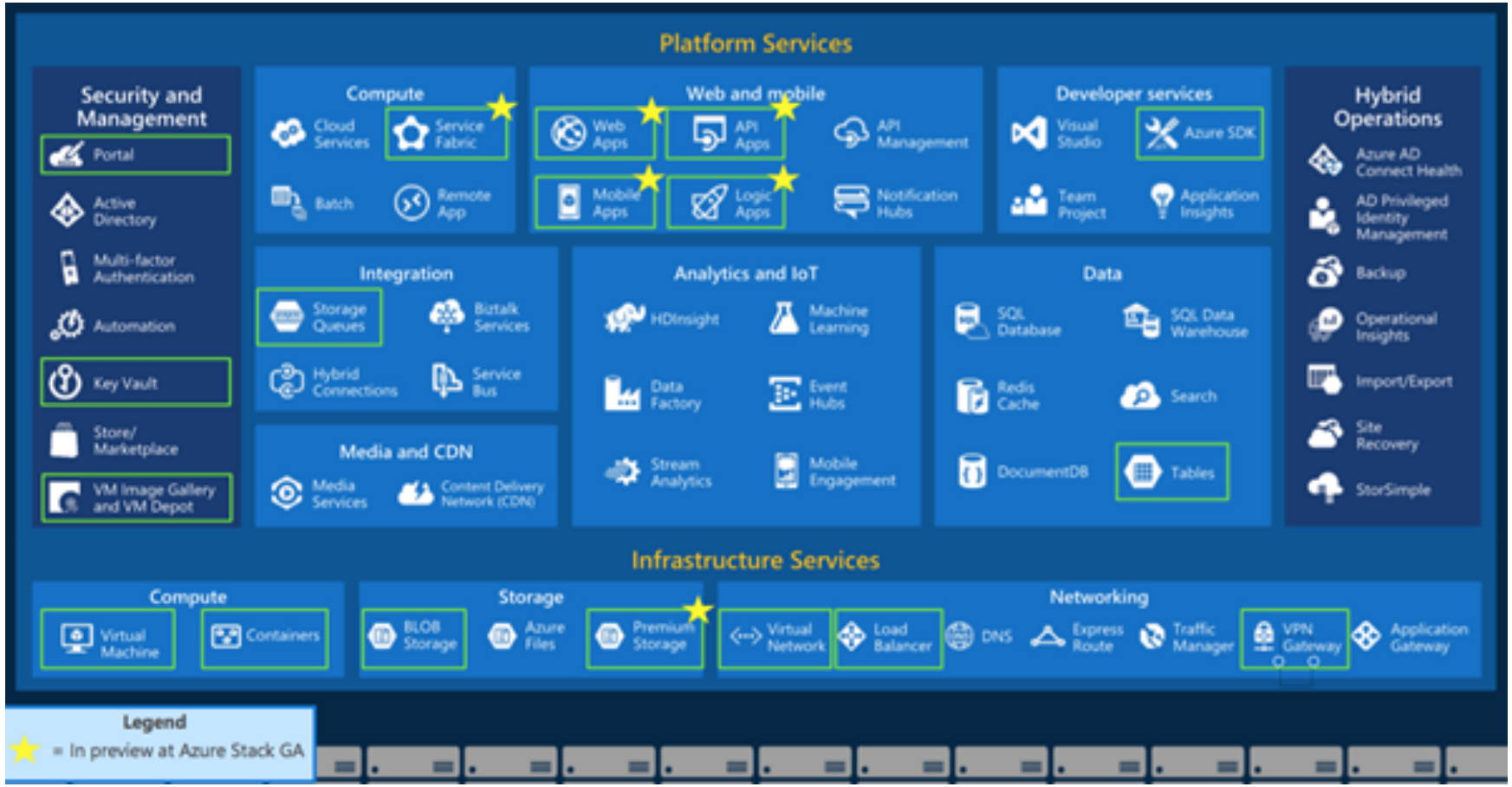
Windows Azure

- Se ajusta a la demanda
- Ciclos y almacenamiento disponible bajo solicitud a un coste
- Se tiene que usar la API de Azure para trabajar con la infraestructura ofrecida por Microsoft
- Las características más significativas: web role, worker role , blob[1] storage, table y drive-storage

https://azure.microsoft.com/es-es/pricing/details/app-service/plans/

[1]Un objeto Blob representa un objeto tipo fichero de datos planos inmutables.

Windows Azure



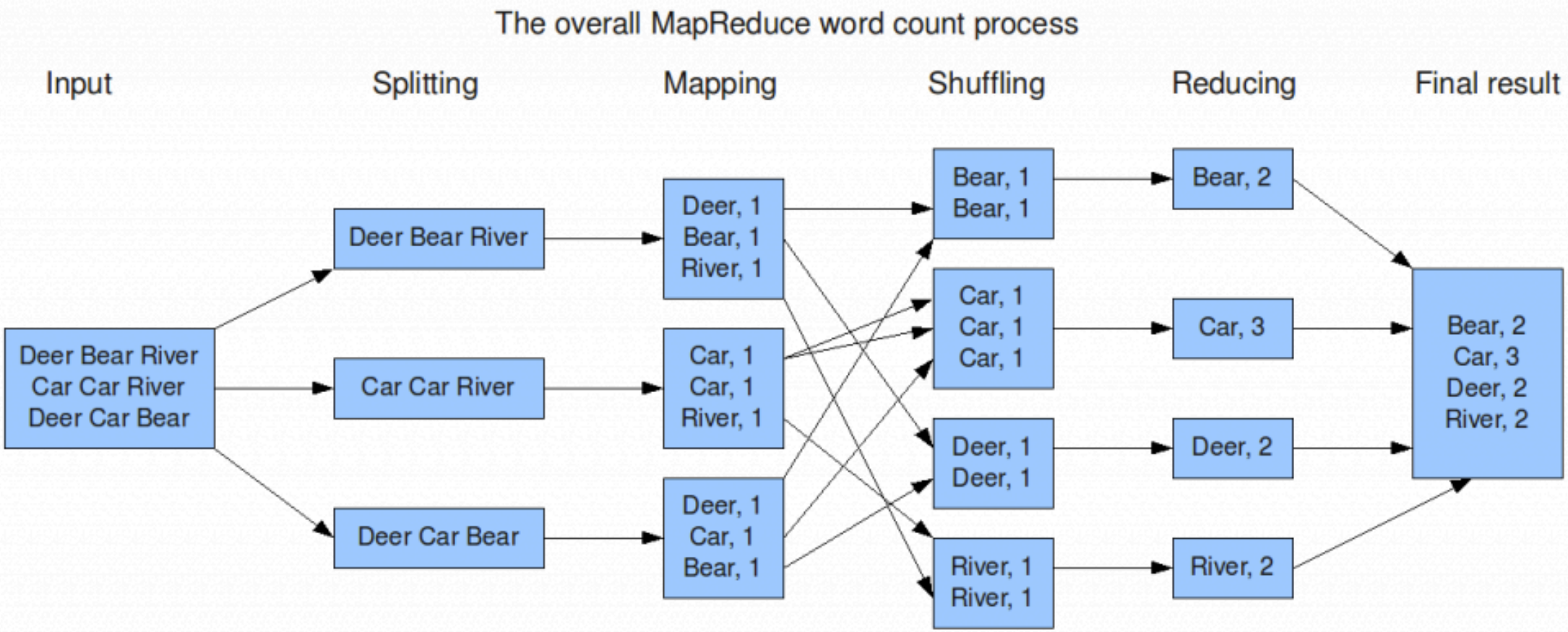
Amazon EC2

- Amazon EC2 es un servicio web.
- EC2 proporciona una API para ejecutar instancias de cualquiera de los SSOO soportados.
- Facilita la computación vía Amazon Machine Images (AMIs) para varios modelos.
- Características: S3, Cloud Management Console, **MapReduce Cloud**, Amazon Machine Image (AMI)
- Excelente distribución, balanceador de carga y herramientas de monitorizacion cloud.

http://aws.amazon.com/es/ec2/pricing/

https://aws.amazon.com/marketplace/ref=mkt_ste_amis_redirect?b_k=291

Amazon EC2


The overall MapReduce word count process
Input
Splitting
Mapping
Shuffling
Reducing
Final result
Deer Bear River
Car Car River
Deer Car Bear
Deer Bear River
Car Car River
Deer Car Bear
Deer, 1
Bear, 1
River, 1
Car, 1
Car, 1
River, 1
Deer, 1
Car, 1
Bear, 1
Bear, 1
Bear, 1
Car, 1
Car, 1
Car, 1
Deer, 1
Deer, 1
River, 1
River, 1
Bear, 2
Car, 3
Deer, 2
River, 2
Bear, 2
Car, 3
Deer, 2
River, 2

Google App Engine

- Ofrece facilidades para el diseño, desarrollo y desplegado de aplicaciones en Java (o casi cualquiera soportado por la JVM), Go and Python.
- Ofrece las mismas características de servicio que en sus propias aplicaciones (PAAS)
- Interface está basado en la programación
- La escala resulta irrelevante (debido al modelo)
- Carcaterísticas: plantillas, excelente monitorización y gestión desde la consola

https://cloud.google.com/products/app-engine/

https://cloud.google.com/products/calculator/

Google App Engine

Google App Engine

- Es una herramienta para el alojamiento de aplicaciones web escalables sobre la infraestructura de Google.
  - Su misión es permitir al desarrollador web crear fácilmente aplicaciones web escalables sin ser un experto en sistemas.
- Admite aplicaciones escritas en varios lenguajes de programación (Java, Python, Ruby, GO).
- Ofrece un servicio de almacenamiento de datos distribuido que incluye un motor de búsqueda y transacciones.
- Se puede utilizar de forma totalmente gratuita.
- Escala fácilmente.

Google App Engine (y1)

- Aporta las siguientes características a los desarrolladores:
  - Limita la responsabilidad del programador al desarrollo y primer despliegue.
    - GAE provee recursos computacionales dinámicamente según son necesarios.
  - Toma control de los picos de tráfico. Si nuestro portal crece en popularidad no es necesario actualizar nuestra infraestructura (servidores, BBDD).
    - Ofrece replicación y balanceo de carga automática apoyado en componentes como Bigtable.
  - Fácilmente integrable con otros servicios de Google. Los desarrolladores pueden hacer uso de componentes existentes y la librería de APIs de Google (email, autenticación, pagos, etc).

GAE: Características

- Ofrece una plataforma completa para el alojamiento y escalado automático de aplicaciones, consistiendo en:
  - Servidores de aplicaciones Python y Java.
  - La base de datos BigTable.
  - El sistema de ficheros Global file System (GFS)
- Como desarrollador simplemente tienes que subir tu código Python o Java compilado a Google, lanzar la aplicación y monitorizar el uso y otras métricas.
- No todas las acciones se permiten (acceso a ficheros, llamadas al SO, algunas llamadas de red).
  - Se ejecuta en un entorno restringido para permitir que las aplicaciones escalen.

GAE: Global file System

- sistema de archivos de Google (GFS), es un sistema de almacenamiento basado en las necesidades de Google
- se basa en las siguientes premisas:
  - El sistema está construido para que el fallo de un componente no le afecte.
  - El sistema almacena grandes archivos
  - La mayoría del trabajo consiste en dos tipos de lecturas: grandes lecturas de datos y pequeñas lecturas aleatorias
  - La carga de trabajo también consiste en añadir grandes secuencias de datos a archivos.
  - El sistema debe ser diseñado para ofrecer concurrencia a múltiples clientes que quieran el mismo archivo.
  - Tener un gran ancho de banda prolongadamente es más importante que una baja latencia.

Facturación GAE

- Hasta 10 aplicaciones con 500 MB de almacenamiento y 5 millones de visitas al mes cada una.

  https://cloud.google.com/appengine/

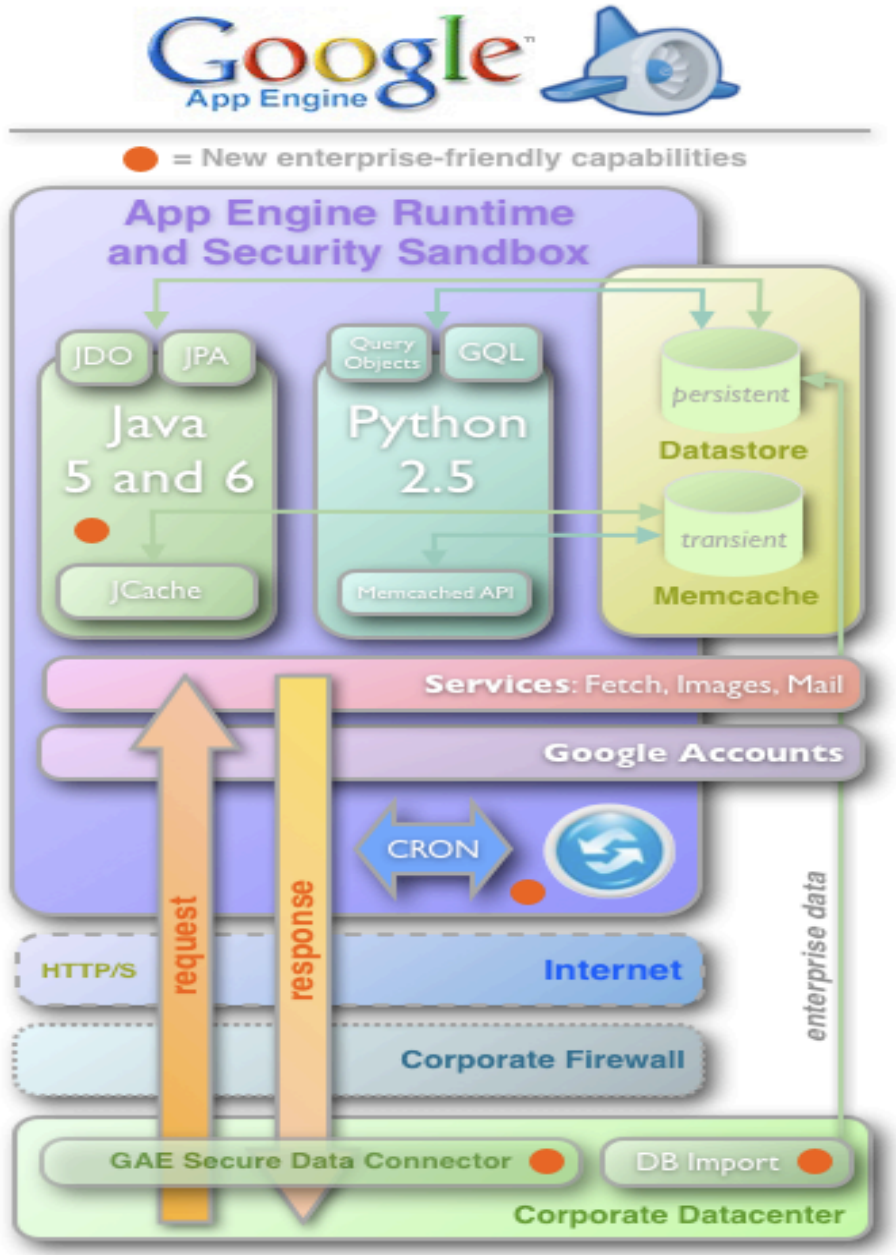
- Página de presupuestado y facturación de recursos:

  https://cloud.google.com/appengine/pricing#costs-for-datastore-calls

- Detalles sobre las cuotas en:

  https://cloud.google.com/products/calculator/

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



From http://blogs.zdnet.com/Hinchcliffe

Limitaciones Google App Engine

- El servicio tiene varias limitaciones:
  - Solo hasta recientemente no todo el mundo podía acceder a él
  - Es gratis durante el periodo de pruebas, pero con límites de uso:
    - Google cobra para webs que requieren alta escalabilidad.
- Existen escasas aplicaciones comerciales desarrolladas en esta plataforma.

  https://cloud.google.com/customers/

  https://cloud.google.com/customers/ubisoft/

Limitaciones Google App Engine (y1)

- Limitaciones técnicas originales parcialmente resueltas:
  - Los desarrolladores solamente tienen acceso de lectura al sistema de ficheros de App Engine.
  - Solamente se puede ejecutar código a partir de una petición HTTP.
  - No se puede descargar o ejecutar scripts en su base de datos (remote_api)
  - Las aplicaciones deben ser escritas en **Python o Java**.

App Engine para Java

- Crea aplicaciones web a través de tecnologías estándar de Java y las ejecuta en la infraestructura escalable Google.
  - Usa JVM Java 6, interfaz de servlets Java y la compatibilidad de interfaces estándar como JDO, JPA, JavaMail y JCache.

- App Engine utiliza el estándar Java Servlet para aplicaciones web.
  - JVM se ejecuta en un entorno seguro de la "zona de pruebas" para aislar tu aplicación por servicio y seguridad.
    - Una aplicación en GAE sólo pueda realizar acciones que no interfieran con el rendimiento ni con la escalabilidad de otras aplicaciones.

Funcionalidad App Engine para Java

- App Engine proporciona un conjunto de servicios escalables que pueden utilizar las aplicaciones para:
  - **Almacenar datos persistentes**. En Java, el almacén de datos admite 2 interfaces Java estándar: los objetos de datos Java (JDO) 2.3 y el API de persistencia de Java (JPA) 1.0.
  - **Acceder a recursos en la red**. A través de la URL Fetch API.
  - **Cachear información**. Memcache de App Engine proporciona un almacenamiento en caché distribuido, transitorio y rápido de los resultados de cálculos y consultas de almacén de datos. La interfaz Java implementa JCache (JSR 107).
  - **Enviar email.** Da soporte de JavaMail para el envío de correos

Funcionalidad App Engine para Java (y1)

- App Engine proporciona un conjunto de servicios escalables que pueden utilizar las aplicaciones para (cont.):
  - **Procesar imágenes**. A través de la Images Java API, permite a las aplicaciones transformar y manipular datos de imágenes en varios formatos.
  - **Gestionar usuarios**. A través de la Users Java API permite utilizar Cuentas de Google para la autenticación del usuario.
  - **Lanzar tareas planificadas o en background**. Mediante la Task Queue Java API y la gestión de tareas por Cron.

Instalación App Engine para Java

1. Descargar el fichero de:
   https://developers.google.com/appengine/downloads?hl=es

2. Descomprimir el fichero .zip.

3. Crear una variable de entorno `APPENGINE_JAVA_SDK` que apunte al directorio raíz de instalación de la SDK.

4. Incluir el directorio `%APPENGINE_JAVA_SDK%\bin` en la variable de entorno `PATH`.

Pasos para crear una aplicación con GAE para Java

1. Crear el proyecto de la aplicación.
2. Crear la clase servlet.
3. Crear el fichero de despliegue de la aplicación: `web.xml`
4. Crear el archivo `appengine-web.xml`
5. Ejecutar el proyecto.
6. Probar el proyecto: http://localhost:8080/<nombre-aplicación>
7. Subir la aplicación al dominio `appspot.com`

Configuración del Entorno

1. Descomprimir la distribución de GAE for Java.
2. Modificar la variable de entorno `APPENGINE_JAVA_SDK` para que apunte a ese directorio.
3. Modificar la variable de entorno `PATH` para que apunte a `%APPENGINE_JAVA_SDK%\bin`
4. Descomprimir el fichero `downloads\apache-ant-1.8.1-bin.zip`
5. Modificar la variable de entorno `PATH` para que apunte a `<ANT_DIR>\bin`

Paso 1: Creando la estructura del proyecto

- Usar el plug-in para Eclipse:
  https://developers.google.com/eclipse/
- Las aplicaciones Java de App Engine utilizan el API Java Servlet para interactuar con el servidor web.
- La estructura del directorio de trabajo será:

```
Guestbook/
  src/
     ...Java source code...
     META-INF/
        ...other configuration...
  war/
    ...JSPs, images, data files...
    WEB-INF/
       ...app configuration...
    classes/
       ...compiled classes...
    lib/
       ...JARs for libraries...
```

src/ contiene el código fuente Java, mientras que otro subdirectorio llamado
war/ contiene la aplicación completa organizada en el formato WAR (webarchive)

Paso 2: Creando la clase Servlet

- Un servlet HTTP es una clase de aplicación que **puede procesar y responder solicitudes web**

- Crear en el directorio `src/guestbook/` un fichero denominado `GuestbookServlet.java` con el siguiente contenido:

```
package guestbook;

import java.io.IOException;
import javax.servlet.http.*;

public class GuestbookServlet extends HttpServlet {
   public void doGet(HttpServletRequest req,
HttpServletResponse resp) throws IOException {
      resp.setContentType("text/plain");
      resp.getWriter().println("Hello, world");
   }
}
```

Paso 3: Creando el fichero de despliegue

- Cuando el servidor web recibe una solicitud, decide qué clase de servlet ejecutar mediante un archivo de configuración conocido como "descriptor de implementación de la aplicación web".
  - Este archivo se denomina **`web.xml`** y se ubica en el directorio `war/WEB-INF/` del directorio que contiene los ficheros de una aplicación web en Java

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE web-app PUBLIC
 "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
 "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app xmlns="http://java.sun.com/xml/ns/javaee" version="2.5">
    <servlet>
        <servlet-name>guestbook</servlet-name>
        <servlet-class>guestbook.GuestbookServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>guestbook</servlet-name>
        <url-pattern>/guestbook</url-pattern>
    </servlet-mapping>
    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
    </welcome-file-list>
</web-app>
```

Paso 4: Crear el fichero de configuración de aplicación GAE

- App Engine necesita un archivo de configuración adicional para poder desarrollar y ejecutar la aplicación, denominado **`appengine-web.xml`**
  - Se ubica en `WEB-INF/` junto a `web.xml`.
  - Incluye la **ID registrada** de la aplicación, el número de versión de la aplicación y listas de archivos que se deben tratar como archivos estáticos (por ejemplo, imágenes y CSS) y archivos de recursos (por ejemplo, JSP y otros datos de aplicación).
- El directorio `war/WEB-INF/` incluye un archivo denominado `appengine-web.xml` que contiene lo siguiente:

```
<?xml version="1.0" encoding="utf-8"?>
<appengine-web-app xmlns="http://appengine.google.com/ns/1.0">
    <application>librocitas</application>
    <version>1</version>
</appengine-web-app>
```

Paso 5: Ejecución del Proyecto

- El SDK de App Engine incluye un servidor web de pruebas para depurar tu aplicación.
- El servidor simula los servicios y el entorno App Engine, que incluyen restricciones en la zona de pruebas, el almacén de datos y los servicios.
- Con el fichero ant ejecuta: `ant runserver`
  - Puedes detenerlo con Ctrl-C

https://developers.google.com/appengine/docs/java/tools/ant?hl=es

https://ant.apache.org/bindownload.cgi

Paso 6: Subiendo la aplicación

1. Puedes crear y administrar aplicaciones web App Engine con la consola de administración de App Engine a través de la siguiente URL: http://appengine.google.com/

2. Para crear una nueva aplicación, haz clic en el botón "Create Application" (Crear aplicación)

3. Edita el archivo `appengine-web.xml` y, a continuación, cambia el valor del elemento `<application>` para que sea la ID registrada de tu aplicación (`librocitas`).

4. Ejecuta el siguiente comando en línea de comandos para subir la aplicación:
   ```
   $ appcfg update www
   ```

5. Vete a: http://librocitas.appspot.com/

Usando el Servicio de Usuarios

- Google App Engine ofrece varios servicios útiles basados en la infraestructura de Google a los que se puede acceder a través de aplicaciones utilizando una serie de bibliotecas incluidas en el kit de desarrollo de software (SDK)
  - Por ejemplo, el **servicio de usuarios** te permite integrar tu aplicación con cuentas de usuarios de Google.

Usando el Servicio de Usuarios

- Google App Engine ofrece varios servicios útiles basados en la infraestructura de Google a los que se puede acceder a través de aplicaciones utilizando una serie de bibliotecas incluidas en el kit de desarrollo de software (SDK)
  - Por ejemplo, el servicio de usuarios te permite integrar tu aplicación con cuentas de usuarios de Google

```
package guestbook;
import java.io.IOException;
import javax.servlet.http.*;
import com.google.appengine.api.users.User;
import com.google.appengine.api.users.UserService;
import com.google.appengine.api.users.UserServiceFactory;

public class GuestbookServlet extends HttpServlet {
    public void doGet(HttpServletRequest req, HttpServletResponse resp)
                throws IOException {
        UserService userService = UserServiceFactory.getUserService();
        User user = userService.getCurrentUser();

        if (user != null) {
            resp.setContentType("text/plain");
            resp.getWriter().println("Hello, " + user.getNickname());
        } else {
            resp.sendRedirect(userService.createLoginURL(req.getRequestURI()));
        }
    }
}
```

Uso de un JSP

- Aunque podríamos generar el código HTML para la interfaz de usuario directamente a partir del código Java del servlet, no sería algo fácil de mantener, ya que el código HTML se complica.

- Es más conveniente utilizar un sistema de plantillas, en el que la interfaz de usuario esté diseñada e implementada en **archivos independientes** con marcadores y lógica para insertar datos proporcionados por la aplicación.

- Al cargar una JSP (Java Server Pages) por primera vez, el servidor de desarrollo lo convierte en código fuente Java y, a continuación, compila este código en código de bytes de Java.

- Al subir la aplicación a App Engine, el SDK compila todas las JSP en código de bytes y únicamente sube el código de bytes.

Logueo de Información con App Engine

- El nuevo servlet utiliza la clase `java.util.logging.Logger` para escribir mensajes en el registro.

- Puedes controlar el comportamiento de esta clase a través de un archivo `logging.properties` y de un conjunto de propiedades del sistema en el archivo `appengine-web.xml` de la aplicación.

- Los ficheros de logueo se descargan con la consola de administración o la aplicación

  `appcfg d`e App Engine: https://appengine.google.com/logs

Uso del Almacén de Datos JDO

- La infraestructura de App Engine se encarga de todas las tareas de distribución, replicación y balanceo de carga de los datos de un API sencilla, además de ofrecer un potente motor de consulta y transacciones.
  - Ofrece dos API: un API estándar y otra de nivel inferior.
  - App Engine for Java permite el uso de dos estándares de API diferentes para el almacén de datos: **Objetos de datos Java (JDO)** y **API de persistencia Java (JPA)**.
    - Estas interfaces las proporciona DataNucleus Access Platform, una implementación de software libre de varios estándares de persistencia Java, con un adaptador para Google DataStore
- Utilizaremos la interfaz **JDO** para la recuperación y la publicación de los mensajes de los usuarios en el almacén de datos de App Engine.
  - Access Platform **necesita un archivo de configuración** que le indique que debe utilizar el almacén de datos de App Engine como servidor para la implementación de JDO: **`META-INF/jdoconfig.xml`**
- Documentación detallada de JDO puede encontrarse en:

https://developers.google.com/appengine/docs/java/datastore/?csw=1

Funcionamiento de JDO

- Al crear clases JDO, debes **utilizar anotaciones Java para describir cómo se deben guardar las instancias en el almacén de datos** y cómo se deben volver a crear al recuperarlas de dicho almacén.
    - Access Platform conecta las clases de datos a la implementación mediante un paso de procesamiento posterior a la compilación, que DataNucleus denomina "mejora" de las clases.
- **JDO permite almacenar objetos Java** (a veces denominados "objetos Java antiguos y simples" o POJO) en cualquier almacén de datos con un adaptador compatible con JDO, como DataNucleus Access Platform
- El complemento Access Platform para el almacén de datos de App Engine permite almacenar instancias de clases definidas en el almacén de datos de App Engine y recuperarlas como objetos mediante API JDO
- Ejemplo: la clase `Greeting` representará mensajes individuales publicados en el libro de invitados de nuestra aplicación

La Clase de Persistencia Greeting

```
package guestbook;

import java.util.Date;

import javax.jdo.annotations.IdGeneratorStrategy;

import javax.jdo.annotations.IdentityType;

import javax.jdo.annotations.PersistenceCapable;

import javax.jdo.annotations.Persistent;

import javax.jdo.annotations.PrimaryKey;

import com.google.appengine.api.users.User;

@PersistenceCapable(identityType = IdentityType.APPLICATION)
```

```
public class Greeting {
  @PrimaryKey
  @Persistent(valueStrategy = IdGeneratorStrategy.IDENTITY)
  private Long id;
  @Persistent
  private User author;
  @Persistent
  private String content;
  @Persistent
  private Date date;
  public Greeting(User author, String content, Date date) {
    this.author = author;
    this.content = content;
    this.date = date;
  }
  ...
}
```

La Clase de Persistencia Greeting

- Esta sencilla clase define tres propiedades para un saludo: `author`, `content` y `date`
- Estos tres campos privados presentan la anotación `@Persistent`, que indica a DataNucleus que debe almacenarlos como propiedades de objetos en el almacén de datos de App Engine.
- La clase también define un campo llamado `id`, una clave `Long` que presenta dos anotaciones: `@Persistent` y `@PrimaryKey`.
- El almacén de datos de App Engine tiene una noción de las claves de entidades y puede representar las claves de varias formas en un objeto.
- Más información sobre cómo definir modelos de datos: https://developers.google.com/appengine/docs/java/datastore/jdo/dataclasses

Usando Ficheros Estáticos

- Hay muchos casos en los que querrás mostrar los archivos estáticos directamente en el navegador web: imágenes, vídeos
- Para una mayor eficiencia, App Engine muestra los archivos estáticos desde servidores independientes en lugar de los que ejecutan servlets.
- **App Engine considera todos los archivos del directorio WAR como archivos estáticos**, salvo JSP y los archivos de `WEB-INF/`
- Cualquier solicitud de una URL cuya ruta coincida con un archivo estático lo muestra
- Puedes configurar los archivos que quieres que App Engine considere como archivos estáticos a través del archivo `appengine-web.xml`
  - La siguiente página da más información al respecto:

  https://developers.google.com/appengine/docs/java/config/appconfig
- Para este ejemplo:
  - Crear `main.css` con el siguiente contenido:

    ```
    body { font-family: Verdana, Helvetica, sans-serif; background-color: #FFFFCC; }
    ```
  - Añadir a `guestbook.jsp` lo siguiente:

    ```
    <head> <link type="text/css" rel="stylesheet" href="/stylesheets/main.css" /> </head>
    ```

Creando de Objetos y Claves

- Para almacenar un objeto de datos sencillo en el almacén de datos, ejecuta el método `makePersistent()` del `PersistenceManager` y transfiérelo a la instancia.

```
PersistenceManager pm = PMF.get().getPersistenceManager();
Employee e = new Employee("Alfred", "Smith", new Date());
try {
   pm.makePersistent(e);
} finally {
   pm.close();
}
```

- Las claves más sencillas están basadas en los tipos `Long` o `String`, pero también se pueden crear con la clase `Key`.

```
import com.google.appengine.api.datastore.Key;
import com.google.appengine.api.datastore.KeyFactory;
// ...
Key k = KeyFactory.createKey(Employee.class.getSimpleName(),
  "Alfred.Smith@example.com");
```

- Para recuperar un elemento por clave podemos usar lo siguiente, se puede pasar como segundo argumento una clave, un entero o un string:

```
Employee e = pm.getObjectById(Employee.class, "Alfred.Smith@example.com");
```

Actualización y Borrado de Objetos

- El siguiente código muestra cómo actualizar un objeto persistente:

```
public void updateEmployeeTitle(User user, String newTitle) {
    PersistenceManager pm = PMF.get().getPersistenceManager();
    try {
        Employee e = pm.getObjectById(Employee.class, user.getEmail());
        if (titleChangeIsAuthorized(e, newTitle) {
            e.setTitle(newTitle);
        } else {
            throw new UnauthorizedTitleChangeException(e, newTitle);
        }
    } finally {
        pm.close();
    }
}
```

- El siguiente ejemplo muestra cómo borrar un objeto:

```
pm.deletePersistent(e);
```

Realizando Consultas con JDO

- JDOQL es similar a SQL, aunque es más adecuado para bases de datos relacionadas con objetos, como, por ejemplo, el almacén de datos de App Engine.
- Dos usos diferentes:
  1. Puedes especificar parte o la totalidad de la consulta mediante métodos de ejecución en el objeto de consulta

```
import java.util.List;
import javax.jdo.Query;
// ...
Query query = pm.newQuery(Employee.class);
query.setFilter("lastName == lastNameParam");
query.setOrdering("hireDate desc");
query.declareParameters("String lastNameParam");

try {
   List<Employee> results = (List<Employee>) query.execute("Smith");
   if (results.iterator().hasNext()) {
      for (Employee e : results) {
      // ...
      }
   } else {
      // ... no results ...
   }
} finally {
   query.closeAll();
}
```

Realizando Consultas con JDO

2. Puedes especificar una consulta completa en una cadena mediante la sintaxis de cadena JDOQL:

```
Query query = pm.newQuery("select from Employee " +
      "where lastName == lastNameParam " +
      "order by hireDate desc " +
      "parameters String lastNameParam");
List<Employee> results = (List<Employee>) query.execute("Smith");
```

3. Otro modo:

```
Query query = pm.newQuery(Employee.class,
      "lastName == lastNameParam order by hireDate desc");
query.declareParameters("String lastNameParam");
List<Employee> results = (List<Employee>) query.execute("Smith");
```

```
Query query = pm.newQuery(Employee.class,
      "lastName == 'Smith' order by hireDate desc");
```

Filtros y Restricciones en Consultas JDO sobre App Engine

- Algunos ejemplos de filtros son:

```
query.setFilter("lastName == 'Smith' && hireDate
  > hireDateMinimum");
query.declareParameters("Date hireDateMinimum");
Query query = pm.newQuery(Employee.class,
      "(lastName == 'Smith' || lastName ==
  'Jones')" +
      " && firstName == 'Harold’”);
```

- ATENCIÓN: importantes restricciones en las consultas, revisar:
  - https://cloud.google.com/appengine/docs/java/datastore/queries
  - Ejemplo: Los filtros de desigualdad sólo están permitidos en una propiedad

Ejemplo Objeto Serializable

- La siguiente clase define un objeto que puede serializarse en JDO:

```
import java.io.Serializable;
public class DownloadableFile implements Serializable {
   private byte[] content;
   private String filename;
   private String mimeType;
   // ... accessors ...
}
```

- La siguiente clase define cómo usarlo:

```
import javax.jdo.annotations.Persistent;
import DownloadableFile;
// ...
@Persistent(serialized = "true");
private DownloadableFile file;
```

Características Avanzadas de Google App Engine

- Planificación de tareas con Cron for Java
  - http://code.google.com/appengine/docs/java/config/cron.html
- Memcache Java API
  - http://code.google.com/appengine/docs/java/memcache/overview.html
- URL Fech Java API
  - http://code.google.com/appengine/docs/java/urlfetch/overview.html
- Envío de mensajes instantáneos con XMPP e email
  - http://code.google.com/appengine/docs/java/xmpp/overview.html
- Colas de tareas – permite ejecutar asíncronamente tareas
  - http://code.google.com/appengine/docs/java/taskqueue/overview.html

Planificación de Tareas con Cron

- El servicio App Engine Cron Service permite planificar tareas que se ejecutan en un momento o periodos determinados.
  - Los trabajos cron (cron jobs) son ejecutados por App Engine Cron Service
- Algunos ejemplos de uso serían:
  - Envío de email con informe diario
  - Actualización de tu caché de datos cada 10 minutos
- Documentación en: http://code.google.com/appengine/docs/java/config/cron.html
  - Formato de las planificaciones:

```
("every"|ordinal) (days) ["of" (monthspec)] (time)
```

Planificación de Tareas con Cron

- El fichero `WEB-INF\cron.xml` controla cron para tu aplicación:

```
<?xml version="1.0" encoding="UTF-8"?>
<cronentries>
  <cron>
    <url>/recache</url>
    <description>Repopulate the cache every 2 minutes</description>
    <schedule>every 2 minutes</schedule>
  </cron>
  <cron>
    <url>/weeklyreport</url>
    <description>Mail out a weekly report</description>
    <schedule>every monday 08:30</schedule>
    <timezone>America/New_York</timezone>
  </cron>
</cronentries>
```

Memcache Java API

- Las aplicaciones web escalables de alto rendimiento utilizan a menudo una caché distribuida de datos integrados en memoria delante o en lugar de un sistema de almacenamiento complejo permanente para algunas tareas
    - App Engine incluye un servicio de memoria caché
- El API Java de Memcache implementa la interfaz JCache (`javax.cache`), un estándar en formato borrador descrito en JSR 107
    - JCache proporciona una interfaz en forma de mapa para recopilar datos
        - Puedes almacenar y recuperar valores de la memoria caché mediante las claves
        - Controlar cuándo los valores vencen en la memoria caché
        - Inspeccionar el contenido de la memoria caché y obtener estadísticas sobre ella
        - Utilizar "funciones de escucha" para añadir un comportamiento personalizado al establecer y borrar valores.

URL Fetch Java API

- GAE permite **realizar conexiones HTTP y HTTPS** a través del servicio URL Fetch, que en el caso de GAE for Java se implementa mediante la clase `java.net.URLConnection`
- La funcionalidad que da es:
  - Acceso sencillo a los contenidos de una página mediante `java.net.URL` y el método `openStream()`
  - El método `openConnection()` de `java.net.URL` devuelve una instancia de `HttpURLConnection`, sobre la que se puede hacer `getInputStream`() y `getOutputStream()`
  - Se pueden cambiar propiedades de la conexión como:
    - Añadir cabeceras: `connection.setRequestProperty("X-MyApp-Version", "2.7.3");`
    - Modificar el hecho de que las peticiones se redirijan directamente: `connection.setRequestProperty("X-MyApp-Version", "2.7.3");`
- Más detalles en: http://code.google.com/intl/en/appengine/docs/java/urlfetch/usingjavanet.html

Mail Java API

- Una aplicación en App Engine puede enviar mensajes en representación del administrador de la página o de usuarios autorizados con cuentas Google
  - La Mail Service Java API hace uso de `javax.mail`
- Se puede configurar tu aplicación para recibir mensajes en una dirección con el formato `string@appid.appspotmail.com`
- Cuando una aplicación se ejecuta en el servidor de desarrollo, el mensaje enviado se imprime en el log, no se envía

Task Queue API

- Una aplicación Java puede crear una cola configurada en el fichero de configuración `WEB-INF/queue.xml`
- Para encolar una tarea, hay que obtener una instancia de `Queue` usando una `QueueFactory` y luego invocar el método `add()`
  - Puedes obtener una cola por nombre declarada en el fichero `queue.xml` o la cola por defecto con el método `getDefaultQueue()`
  - Se puede añadir una tarea a la cola pasando una instancia de `TaskOptions` al método `add()`
    - Se invocará un servlet que será quien ejecute la tarea encolada
- Espacio de nombres en versión beta: `com.google.appengine.api.labs.taskqueue`

Importación y Exportación de Datos

- Se pueden acceder a datos detrás de tu Intranet desde una aplicación de Google App Engine, con Google Secure Data Connector y el servicio urlfetch (`com.google.appengine.api.urlfetch.*`)
- Se pueden importar y exportar datos del datastore en forma de ficheros CSV
  - Solamente disponible en Python de momento:
    - http://code.google.com/appengine/docs/python/tools/uploadingdata.html

Google App Engine for Business

- Las aplicaciones generadas con App Engine for Business usan las APIs de Java y Python, pero permiten acceder al desarrollador a capacidades especiales (premium):
  - Acceso a SSL y SQL
  - Tendrán un coste adicional
- Mas información en:

http://code.google.com/appengine/business/

Google web toolkit

¿Qué es GWT?

Esencialmente, es un framework para crear aplicaciones AJAX (Asynchronous JavaScript And XML) que usa tecnologías ampliamente extendidas.

https://developers.google.com/web-toolkit/tools/gwtdesigner/tutorials/loginmanager

Características de GWT

- Las aplicaciones GWT pueden tener varias ventanas contenidas bajo un único padre
- Es sencillo de usar (la construcción de UI es similar a la de Swing)
- Reduce costes (hay estimaciones que indican que se desarrolla 5x más rápido que con J2EE)
- No se necesita servidor y, si se usa, la mayoría de las computaciones se pueden delegar al cliente
- Se optimiza el ancho de banda

Ventajas de Ajax

- Se actualiza de manera asíncrona, se actualizan las páginas sin recargas (no hay pantallas en blanco)

- Escala mejor (clientes con estado y servidores sin estado)

Inconvenientes de Ajax

- Compatibilidad entre navegadores
- Fugas de memoria
- Tiempo de carga
- Se necesita Javascript
  - IDE ajax
  - errores (javascript es dinámico)
  - depurador de javascript
  - seguridad
  - …

Manifiesto de GWT

- GWT debe ayudar a crear código estable, eficiente y compatible con múltiples navegadores
- GWT debe ser amigable con los desarrolladores
  - Compatible con IDEs, soportar depurado, refactorización, fuerte tipado,…
- «En primer lugar el usuario, luego el desarrollador» - Bruce Johnson

La solución

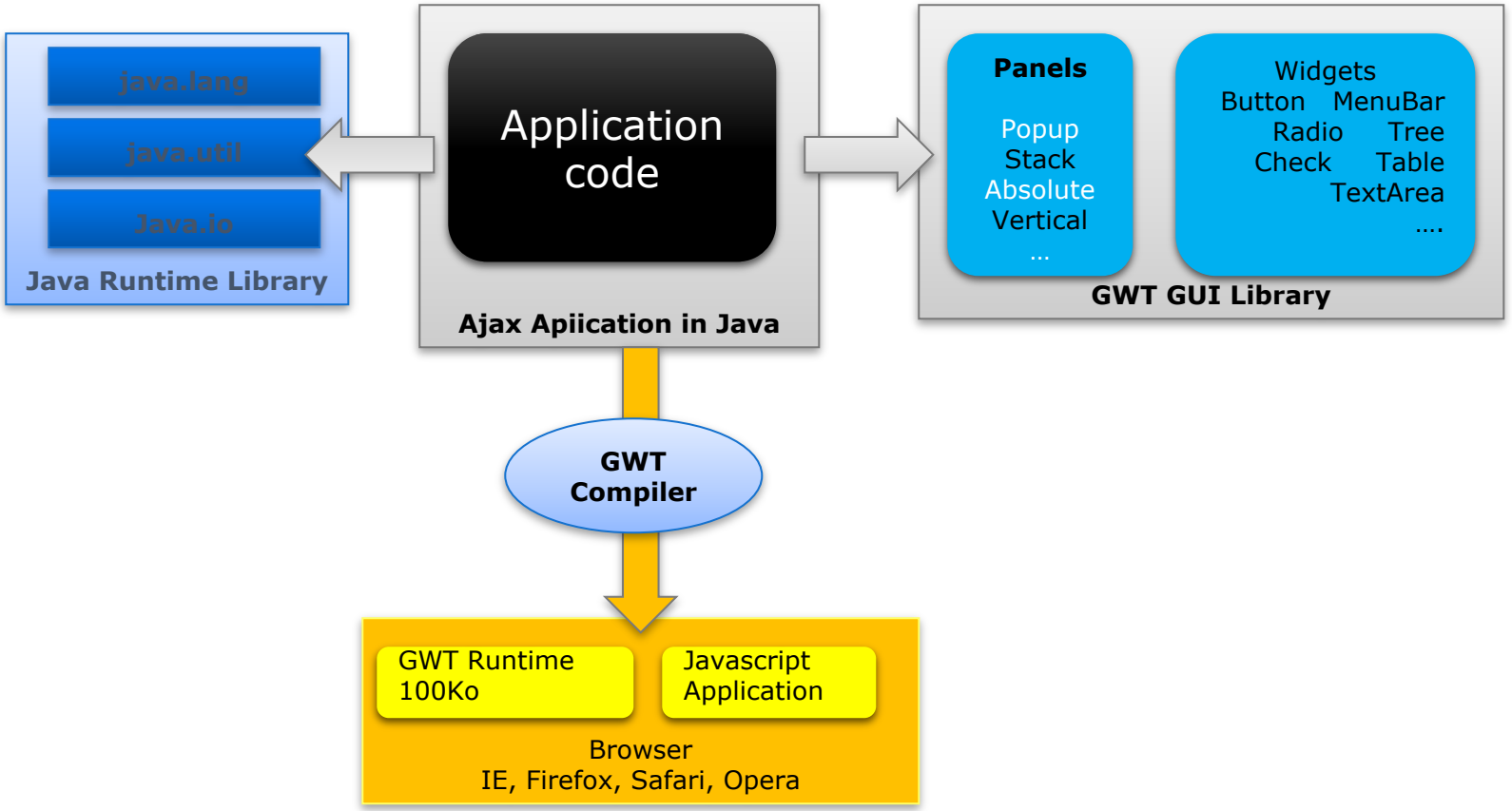
- Codificar en Java
- Compilar el Java a Javascript
  - La compilación se hace partiendo de los AST (Abastract Syntax Tree) de Java, no del bytecode
  - Parte del UI se transforma directamente a HTML
  - Se pueden usar librerías de Javascript desde el código Java usando el JSNI (Javascript Native Interface)

```
public static native void alert(String msg) /*-{
    $wnd.alert(msg);
}-*/;
```

Hello World

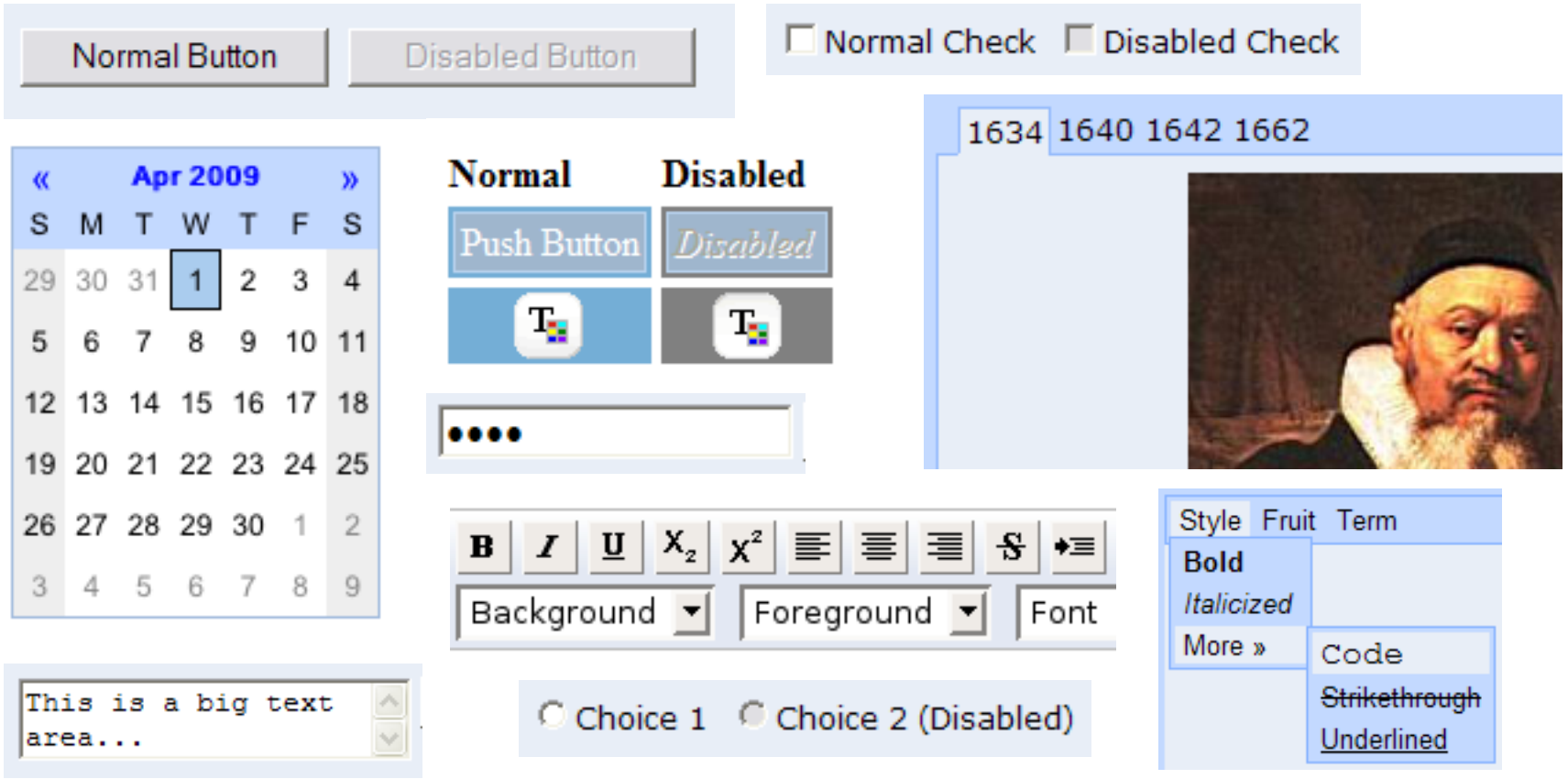
```
public class HelloWorld implements EntryPoint {
    private Button clickMeButton;
    public void onModuleLoad() {
        RootPanel rootPanel = RootPanel.get();

        clickMeButton = new Button();
        rootPanel.add(clickMeButton);
        clickMeButton.setText("Click me!");
        clickMeButton.addClickListener(new ClickListener() {
            public void onClick(Widget sender) {
                Window.alert("Hello, GWT World!");
            }
        });
    }
}
```

Arquitectura de GWT



https://unpocodejava.com/2012/01/12/un-poco-de-gwt/

Widgets de GWT (100% Java)



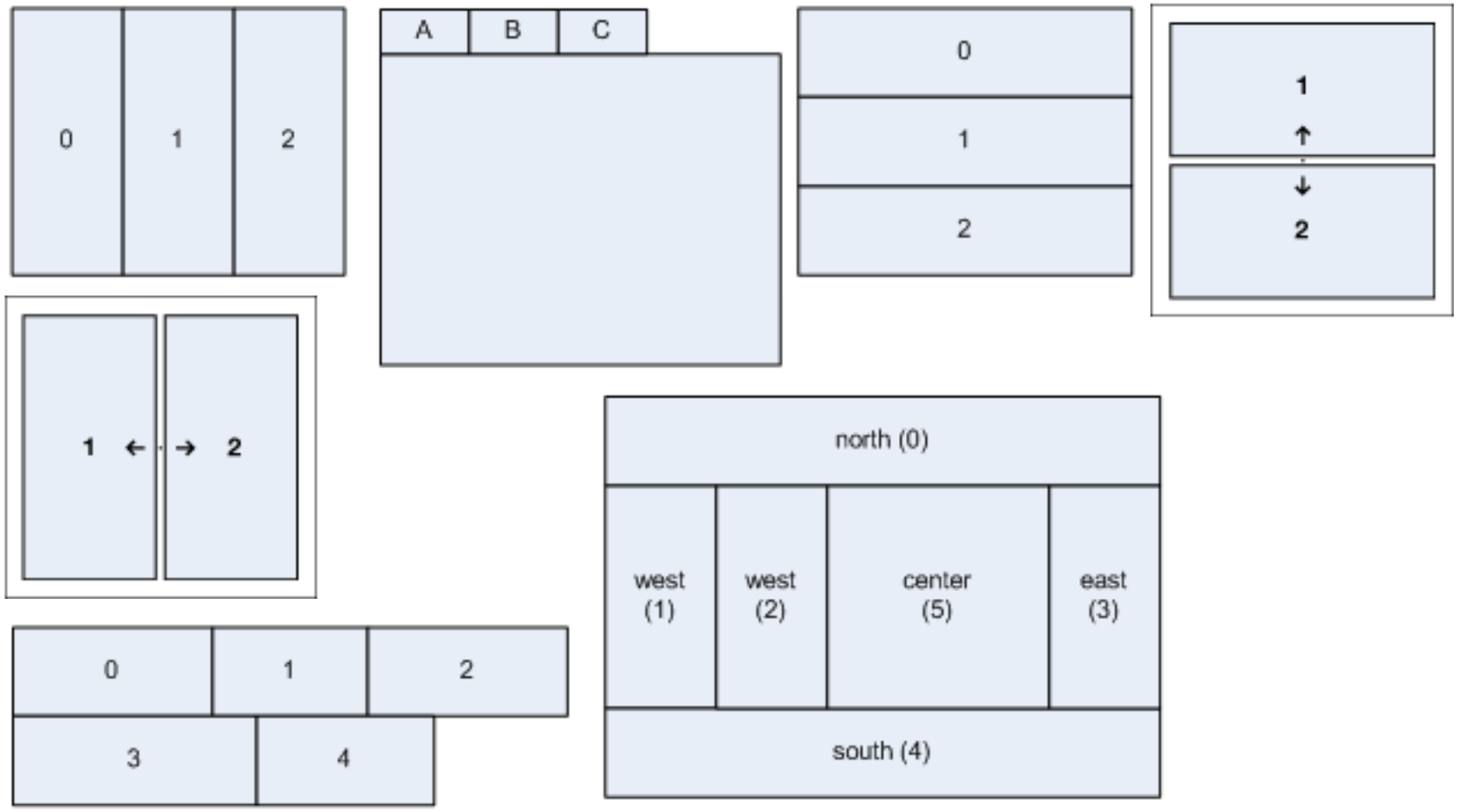
Botones, selectores de fecha, checkbox, áreas de texto, contraseñas, tabs, menus, … Son compatibles con CSS.

Demo en http://www.gwtproject.org/doc/latest/tutorial/index.html

Otros enlaces de interés

- http://www.gwt-ext.com/demo/#combinedLayout
- http://www.smartclient.com/smartgwt/showcase/#main
- https://www.openhub.net/p/google-web-toolkit-incubator
- https://vaadin.com/framework
- https://www.openhub.net/p/gwt-mosaic
- https://examples.javacodegeeks.com/enterprise-java/gwt/gwt-widgets-tutorial/

Paneles



Y muchos más…

Soporta RPC (JSON/XML-RPC)

- JSON-RPC es un protocolo de llamada a procedimiento remoto codificado en JSON.
- JSON-RPC Permite recibir notificaciones (datos enviados al servidor que no requiere de una respuesta) y de varias llamadas a ser enviados al servidor que puede ser contestada fuera de orden.

```
private void getData(){

    DataServiceAsync dataService =
       (DataServiceAsync) GWT.create( DataService.class );

    ServiceDefTarget endpoint = (ServiceDefTarget) dataService;
    endpoint.setServiceEntryPoint("/DataService");

    dataService.getData(new AsyncCallback(){
        public void onSuccess(Object result) {
            table.setSource(
                new SimpleDataSource( (Person[]) result ) );
        }

        public void onFailure(Throwable caught) {
            Window.alert("Unable to get data from server: "
                +caught.toString());
        }

    });
```

Soporta I18N/L10N

- Combinados, los métodos, protocolos y aplicaciones de I18N/L10N permiten a los usuarios **usar el idioma de su elección**

- L10N sigue el mismo esquema, y procede de “localización”

```
******  Déclaration d'une interface
public interface GameStatusMessages extends Messages {
      String turnsLeft(String username, int numTurns);
      String currentScore(int numPoints);
    }

****** Définition d'un fichier de propriétés (localisées)
turnsLeft = Turns left for player ''{0}'': {1}
currentScore = Current score: {0}

****** Utilisation
GameStatusMessages messages =
      (GameStatusMessages) GWT.create(GameStatusMessages.class);
messages.turnsLeft(username, turnsLeft);
```

Múltiples navegadores








Optimización

- El código Javascript es optimizado
- Sólo se descarga el Javascript necesario para el navegador
  - IE descarga sólo el javascript para IE, Firefox descarga sólo el javascript para Firefox,…
  - Las APIs no utilizadas se eliminan del javascript compilado
- Las imágenes se pasan en un único bloque

Herramientas

- GWT está soportado en los principales IDE de Java: **Eclipse**, NetBeans, IntelliJ IDEA, JDeveloper,…
- El Interfaz se puede desarrollar gráficamente con GWT Designer
- Hay depuradores (de hecho son los mismos que en Java)

All together now!