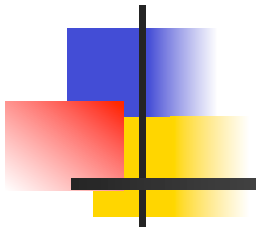
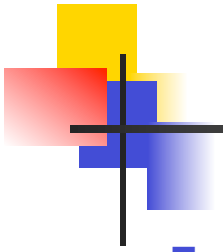
Introducción a AJAX y visión global de la práctica



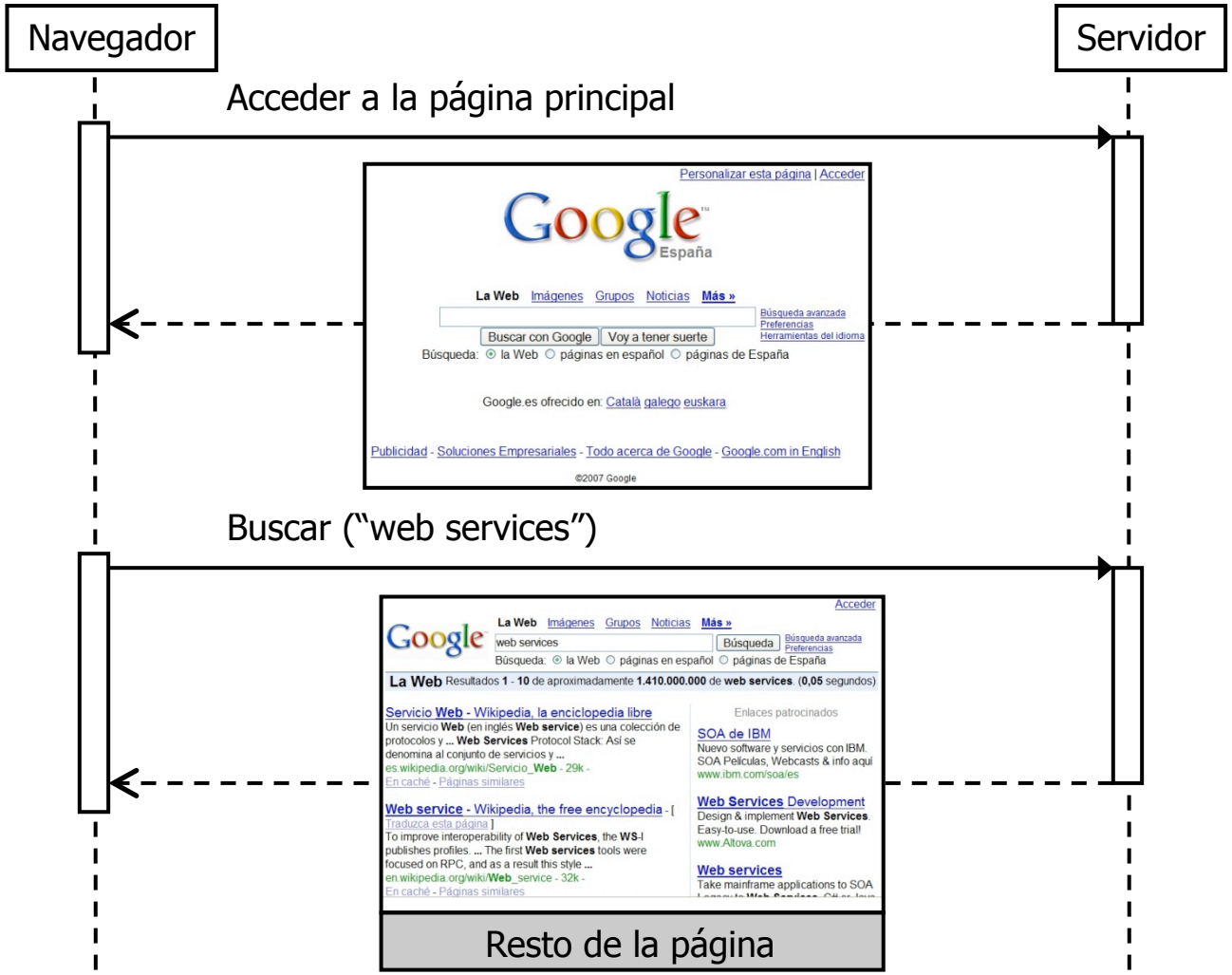
Modelo de aplicaciones Web clásico (1)

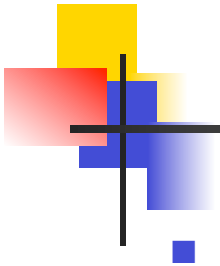


- La mayor parte de las interacciones del usuario causan una petición HTTP al servidor Web
- El servidor Web procesa la petición y devuelve la nueva página a mostrar
  - Internamente la petición ejecuta una lógica de negocio (e.g. búsqueda de productos, añadir un producto al carrito, realizar un pedido, etc.) y devuelve una nueva página HTML con la respuesta
  - Alternativamente la respuesta podría indicar un error o una redirección

Modelo de aplicaciones Web clásico (2)

- Ejemplo: búsqueda en Google

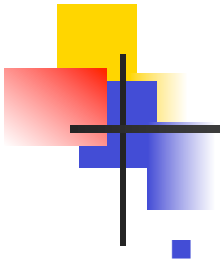




Modelo de aplicaciones Web clásico (y 3)

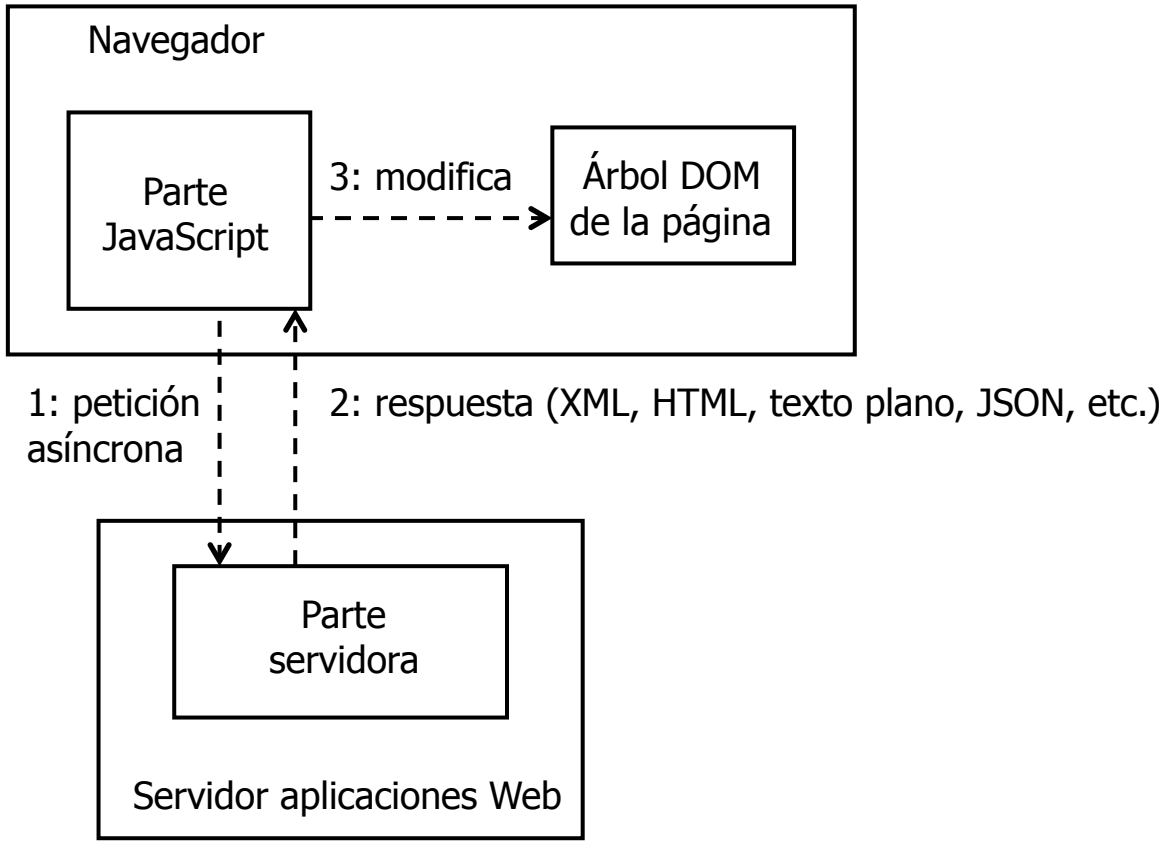
- Ventajas del modelo
  - **Ha sido y es útil**
  - Soportado por numerosas tecnologías maduras => **facilidad de desarrollo**
- Limitaciones del modelo
  - **Las interfaces del usuario son poco interactivas** con respecto a las aplicaciones de escritorio
    - Ejemplo:
      - Una aplicación Web clásica que presenta un formulario, no puede validar los campos (e.g. comprobar que un identificador de login existe) hasta que el usuario pulsa el botón de submit, excepto validaciones simples que se puedan hacer mediante JavaScript
      - Una aplicación de escritorio podría hacerlo cada vez que el usuario rellena un campo, o incluso, mientras lo está rellenando (captura eventos de usuario y valida)
    - Otros ejemplos: no tienen soporte de drag-n-drop, no es posible refrescar la información de una zona de la página sin tener que recargar toda la página, etc.

Modelo de aplicaciones AJAX (1)

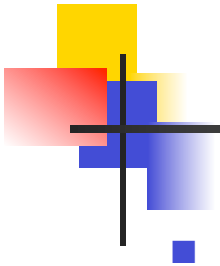


- AJAX es un término acuñado por Jesse James Garrett
  - http://www.adaptivepath.com/publications/essays/archives/000385.php
  - AJAX = Asynchronous JavaScript + XML
- Popularizado por Google
  - Google Maps, Gmail, Google Calendar, etc.
- Es un **enfoque** para la construcción de aplicaciones Web con una interactividad similar a la que presentan las aplicaciones de escritorio
  - La aplicación Web tiene dos partes: parte cliente y parte servidora
  - La parte cliente es en su mayor parte JavaScript
  - La parte servidora recibe peticiones HTTP **asíncronas** procedentes de la parte cliente
  - A pesar de que el acrónimo AJAX sugiere el uso de XML para el envío de datos, **los datos pueden pasarse en el formato que se desee** (e.g. XML, HTML, texto plano, JSON, etc.)

Modelo de aplicaciones AJAX (2)

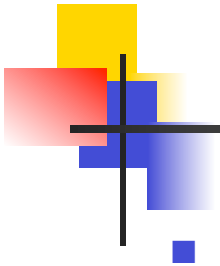


Modelo de aplicaciones AJAX (3)



- Parte cliente
  - Baja el código JavaScript necesario cuando sea preciso (e.g. usando el tag **`<script>`** en una respuesta HTML)
    - NOTA: el navegador usa el API de DOM para representar el HTML de la página que visualiza, y desde JavaScript se puede acceder al API de DOM para manipular el árbol
  - Funcionamiento
    - Captura eventos del usuario (clicks, drap-n-drop, etc.)
    - Si puede resolver el evento localmente, lo hace
    - En otro caso (e.g. una búsqueda en la base de datos), envía una petición asíncrona por HTTP a la parte servidora usando el objeto **`XMLHttpRequest`**
    - Cuando llega la respuesta, modifica el árbol DOM de la página
      - Ejemplo: si la respuesta contiene los resultados de una búsqueda en XML, añade nodos al árbol DOM para presentar los resultados en HTML

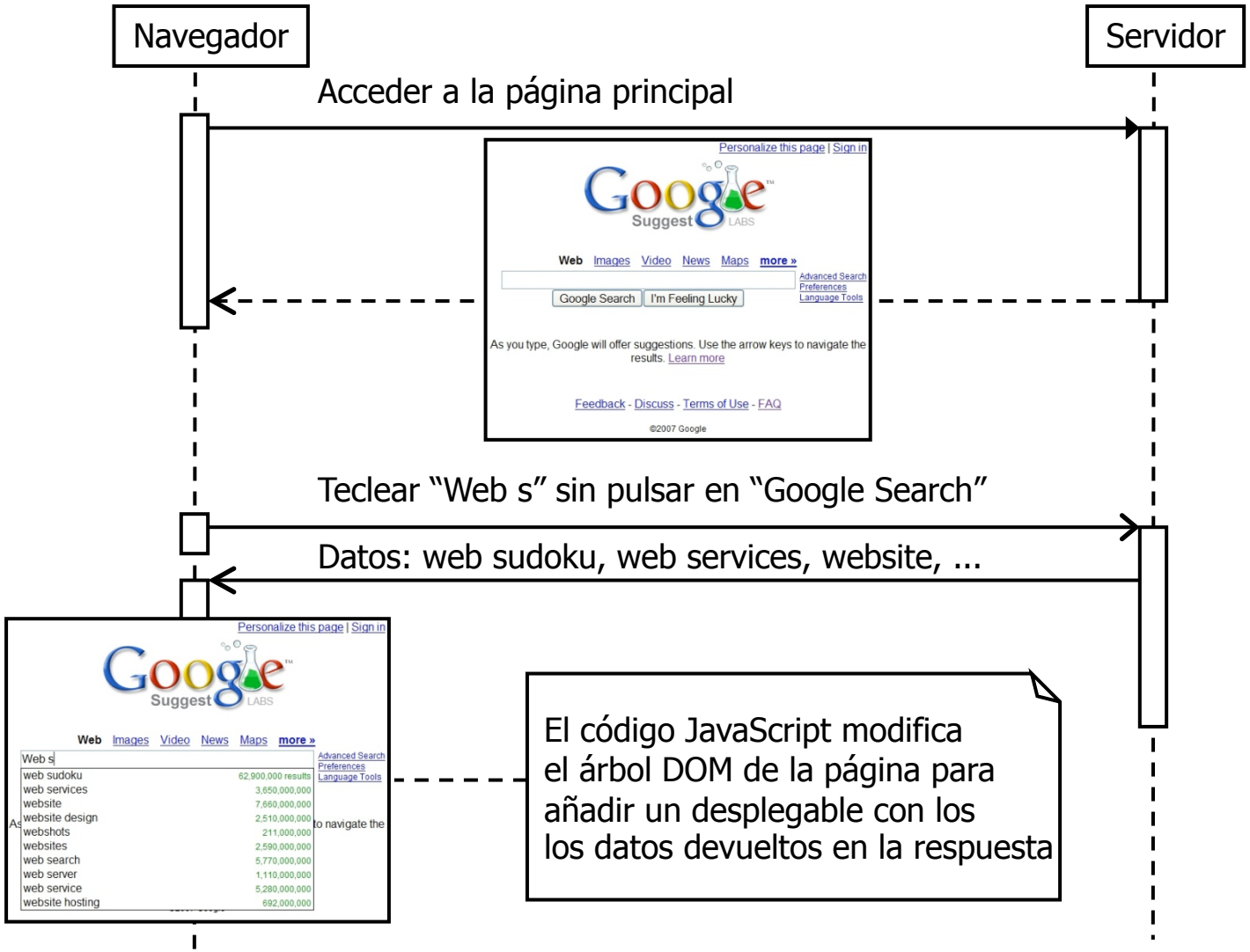
Modelo de aplicaciones AJAX (4)



- Parte cliente (cont)
  - NOTAS
    - Dado que la petición es asíncrona, el navegador no se queda bloqueado, y el usuario "podría" continuar interactuando con la aplicación mientras no llegue la respuesta
      - Sin embargo, esta ventaja muchas veces no se explota, dado que la respuesta llega rápido o no tiene sentido para el usuario hacer algo mientas tanto
    - El código JavaScript suele utilizar estilos CSS en los nodos que añade al árbol DOM
      - El look-n-feel de la aplicación se puede cambiar sin modificar el código JavaScript
- Parte servidora
  - Recibe peticiones HTTP
  - Invoca lógica de negocio (e.g. lanzar una consulta a la base de datos)
  - Devuelve datos

Modelo de aplicaciones AJAX (5)

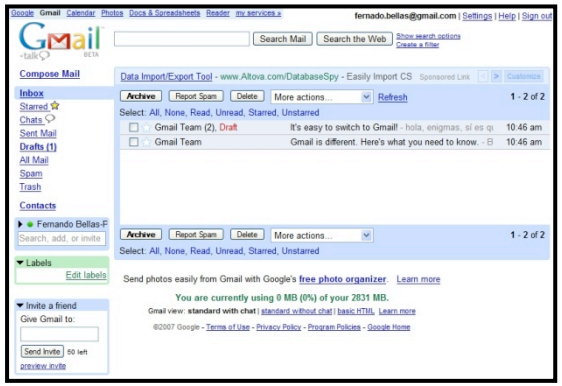
- Ejemplo: búsqueda en Google Suggest



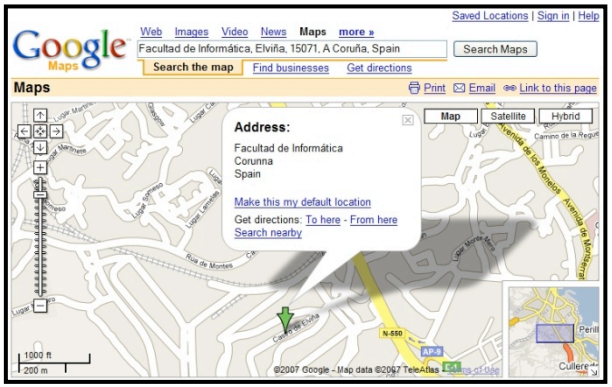
Modelo de aplicaciones AJAX (y 6)

- Actualmente muchas aplicaciones Web usan AJAX para algunos (pocos) casos de uso
  - Aquellos en los que un enfoque clásico resulta poco admisible para el usuario final
- Otras son 100% AJAX

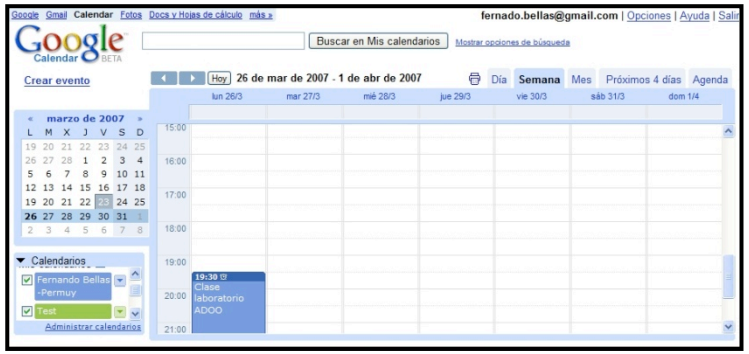
Gmail
(http://gmail.google.com)

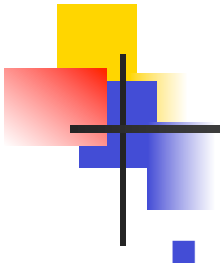


Google Maps
(http://maps.google.com)



Google Calendar
(http://www.google.com/calendar)

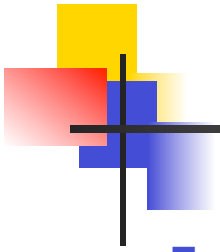




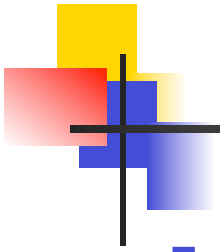
Desarrollo AJAX (1)

- Existen un gran número de APIs para AJAX
  - Soporte para parte cliente: invocaciones asíncronas + manipulación del árbol DOM
  - Soporte para parte servidora: framework para implementar los servicios
- Java
  - DWR (http://getahead.org/dwr)
  - Google Web Toolkit (http://code.google.com/webtoolkit)
  - AjaxTags (http://ajaxtags.sourceforge.net)
  - Las últimas versiones de los frameworks Web (Struts 2, Tapestry 5, Apache Wicket, etc.) incluyen soporte nativo para AJAX
- .NET
  - ASP.NET AJAX (http://ajax.asp.net)
    - Integrado nativamente en la última versión de ASP.NET
  - El framework Web MVC MonoRail (http://www.castleproject.org) incluye soporte para AJAX

Desarrollo AJAX (2)

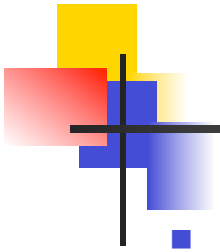


- Ruby
  - El framework Web MVC Ruby on Rails (http://www.rubyonrails.org) incluye soporte para AJAX
- Librerías JavaScript
  - Existen librerías JavaScript para facilitar la implementación de la parte cliente
    - Se pueden usar en combinación con los anteriores frameworks (algunos las integran directamente)
  - Prototype (http://www.prototypejs.org)
  - Script.aculo.us (http://script.aculo.us)



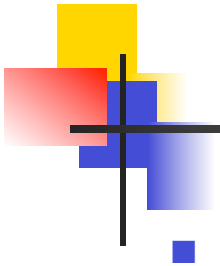
Desarrollo AJAX (y 3)

- Principal inconveniente del enfoque AJAX: dificultad de desarrollo
  - Frameworks poco maduros (cambiantes)
  - Tediosos de usar (bajo nivel)
  - Con la mayoría es necesario escribir código JavaScript
    - Gran esfuerzo para garantizar que funcione en todos los navegadores
    - Código difícil de mantener
- Por este motivo actualmente es más **prudente** usar AJAX sólo para implementar los casos de uso que verdaderamente lo requieren



GWT (1)

- GWT (Google Web Toolkit) es un framework Java para AJAX que permite implementar la **parte cliente y servidora en Java**
  - Objetivo: **facilidad de desarrollo** (el código JavaScript es transparente al desarrollador)
- Parte cliente y servidora se paquetizan en una aplicación Web Java (.war)
  - Se puede instalar en cualquier servidor de aplicaciones Web Java
- Parte servidora
  - Los servicios se especifican en interfaces "normales" Java (con operaciones "normales"), e implementarlos sólo consiste en implementar estos interfaces
  - Internamente las peticiones llegan a un servlet, que delega en la operación del interfaz correspondiente a esa petición y devuelve el resultado
    - GWT utiliza JSON (JavaScript Object Notation) para pasar los parámetros y devolver el resultado
    - JavaScript tiene soporte directo para convertir objetos JavaScript a JSON y viceversa
    - Por este motivo, JSON es muy usado como alternativa a XML en las interacciones AJAX

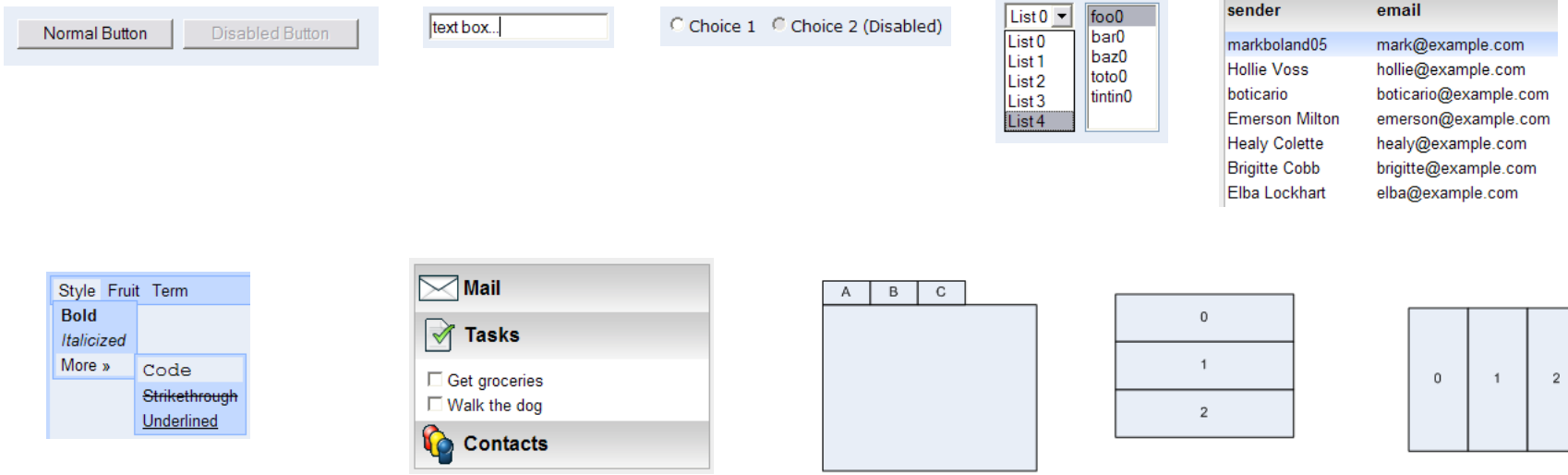


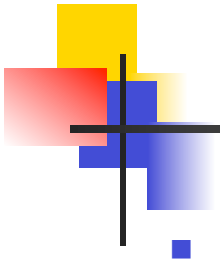
GWT (2)

- Parte cliente
  - GWT incluye un compilador de Java-a-JavaScript
    - Convierte el código Java a JavaScript
    - Código JavaScript eficiente y válido para los navegadores más usuales
  - Restricciones al código Java
    - Actualmente el código fuente tiene que ser compatible con J2SE 1.4.2 (sintaxis de JavaSE 5 no soportada) + ciertas restricciones (e.g. **`Throwable.getStackTrace`** no soportado)
    - Java Runtime Emulation (JRE) library
      - El código cliente sólo puede usar las clases más comunes de **`java.lang`** y **`java.util`**
        - Ej.: no puede usar las de **`java.io`** (e.g. acceso a ficheros locales)
      - La librería JRE genera código JavaScript equivalente
  - Soporte para realizar invocaciones asíncronas (y recibir las respuestas) a los métodos de los interfaces de la parte servidora

GWT (3)

- Parte cliente (cont)
  - Framework orientado a componentes gráficos (widgets)
    - Botones, cajas de texto, selectores, tablas, barras de menú, paneles, etc.
    - Permite crear nuevos componentes



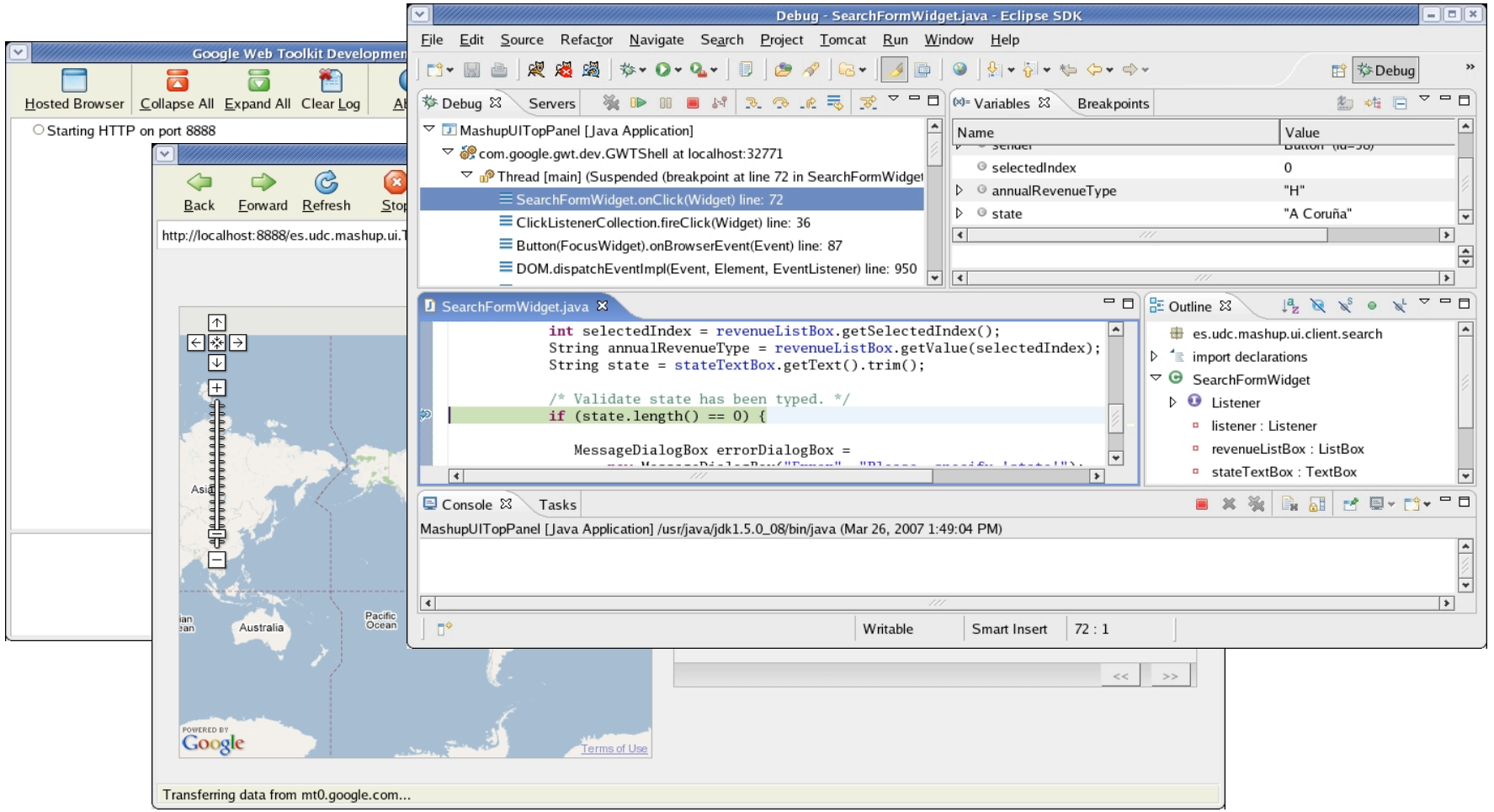


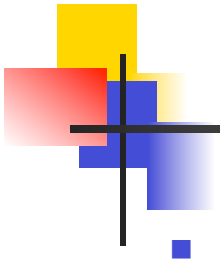
GWT (4)

- Modos de ejecución
  - **Web mode**
    - Modo de ejecución real
    - Aplicación Web Java con lado cliente (JavaScript) y servidor (servlet que recibe peticiones + interfaces + implementación de los interfaces)
    - Como en cualquier aplicación Web Java, la parte servidora se puede depurar desde un IDE que incluya un plugin para el servidor de aplicaciones Web
  - **Hosted mode**
    - GWT incluye una aplicación Java (GWT Shell / GWT Web Browser) que permite ejecutar una aplicación GWT sin generar el JavaScript
      - Internamente es una versión modificada de Tomcat y proporciona un sencillo navegador
    - Como todo es Java (el GWT Shell y la aplicación GWT), se puede ejecutar el GWT Shell desde un IDE en modo depuración, y así depurar tanto la parte cliente como la servidora desde el IDE
      - Facilidad de depuración

GWT (5)

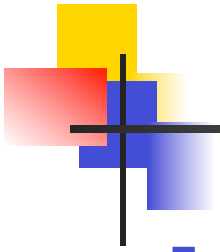
- Modos de ejecución (cont)
  - **Hosted mode** (cont)





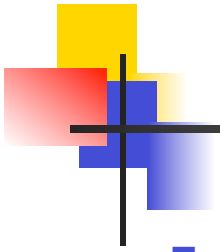
GWT (y 6)

- Estado actual
  - Los widgets que vienen con GWT son muy básicos
    - El framework de GWT pretende proporcionar el soporte necesario para que sobre él se puedan construir widgets más elaborados
  - Empiezan a surgir librerías de widgets más elaboradas
    - GWT Widget Library (http://gwt-widget.sourceforge.net)
  - IDEs para GWT
    - Permiten construir la interfaz gráfica visualmente ("a la Visual Basic")
    - Instantiations GWT Designer (http://www.instantiations.com)
    - JetBrains GWT Studio (http://www.jetbrains.com)
    - Wirelexsoft VistaFei for Google Web Toolkit (http://www.wirelexsoft.com)
- ¿Para qué usar GWT?
  - Para aplicaciones Web 100% AJAX o para partes AJAX significativamente complejas de aplicaciones Web convencionales
  - Para el resto, mejor frameworks más pequeños



Práctica: Mashup

- Desarrollo de una sencilla aplicación de tipo **mashup**
  - "Mashup" es un término que hace referencia a una aplicación que se apoya en un conjunto de servicios remotos (no necesariamente Servicios Web) para proporcionar su funcionalidad.
  - Las aplicaciones Web de tipo mashup son características de la denominada **Web 2.0**, que hace referencia a una serie de elementos que caracterizan a las nuevas aplicaciones Web
    - T. O'Reilly, "What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software", http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html
    - AJAX y REST son también característicos de las aplicaciones Web 2.0

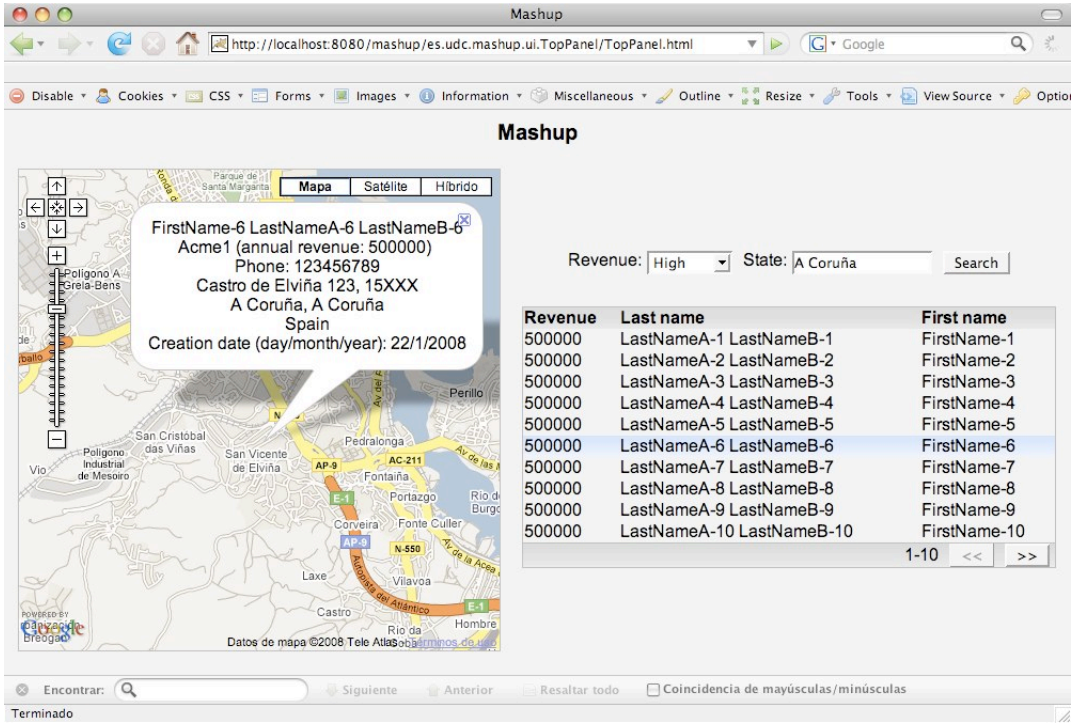


Práctica: contexto (1)

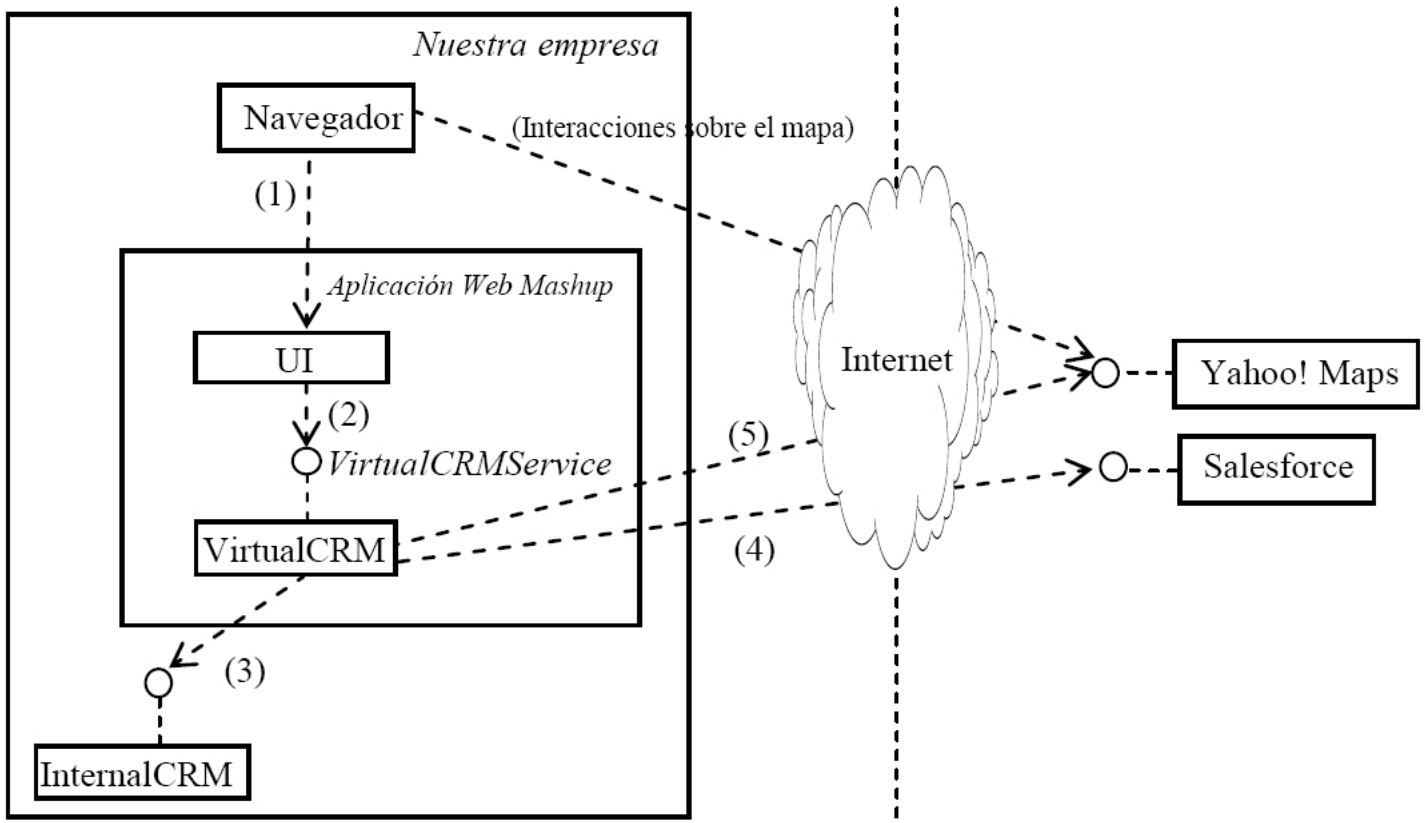
- Se supondrá que trabajamos en una empresa que por razones históricas mantiene la información de sus clientes en dos CRMs
  - Uno interno
    - Instalado en la empresa hace tiempo
  - Otro externo (accesible vía Internet)
    - Nuestra empresa acaba de adquirir otra empresa de un área de negocio afín. Esta empresa había contratado un servicio de CRM con Salesforce (http://www.salesforce.com), de manera que los datos sobre los clientes de esta nueva empresa los gestiona Salesforce.
    - Accesible por usuarios finales mediante aplicación Web
    - Ofrece servicios SOAP para permitir construir aplicaciones que accedan a los datos de los clientes

Práctica: contexto (y 2)

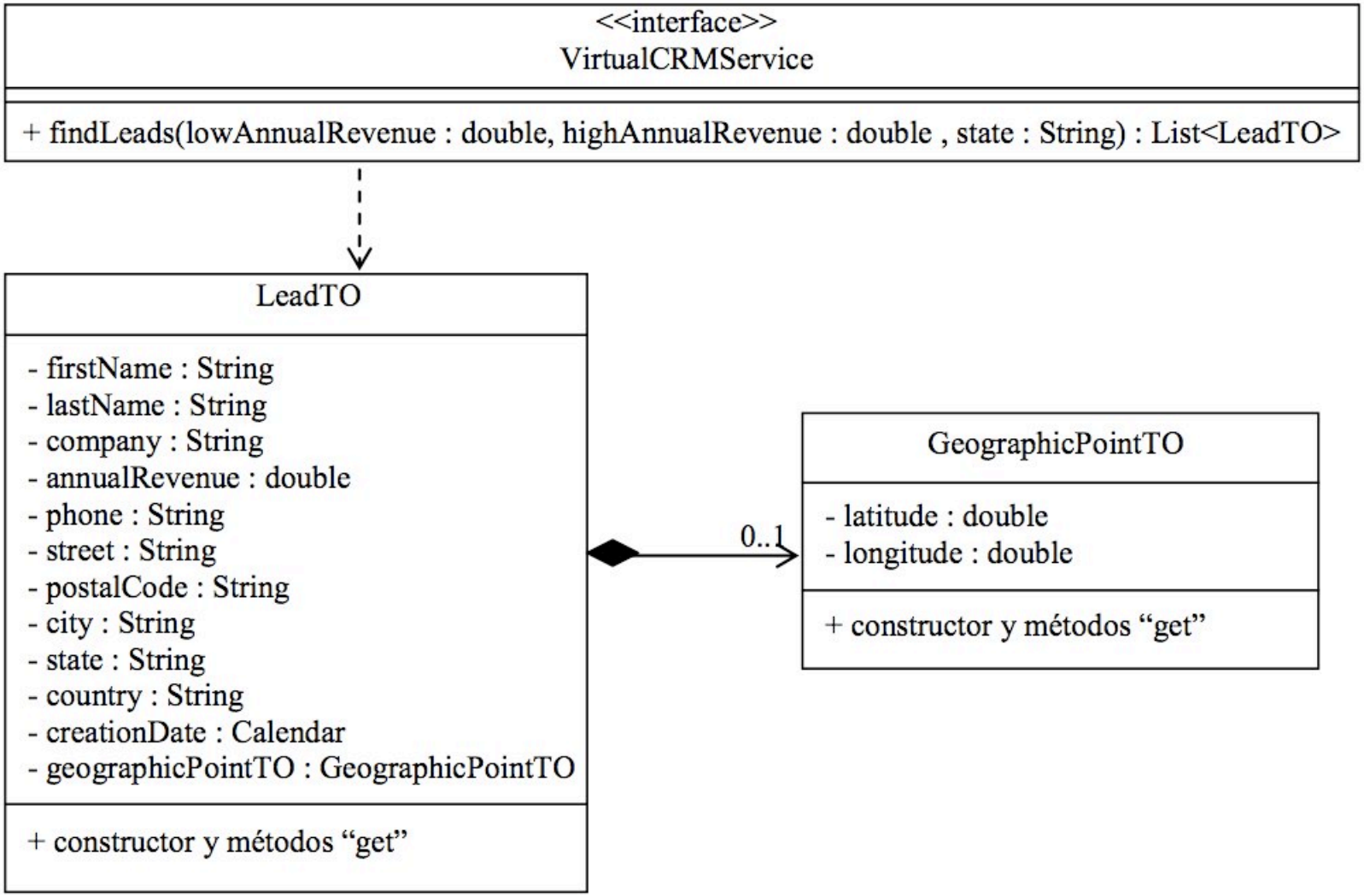
- La empresa se encuentra con que tiene datos sobre clientes en dos CRMs: en el interno y en Salesforce
- Se desea construir una aplicación que permite recuperar la información de los clientes de tipo "Lead" (clientes potenciales) de manera **unificada**
  - Además la información de los clientes incluirá la latitud y longitud de los clientes, de manera que la aplicación pueda posicionarlos sobre un mapa (Google Maps)



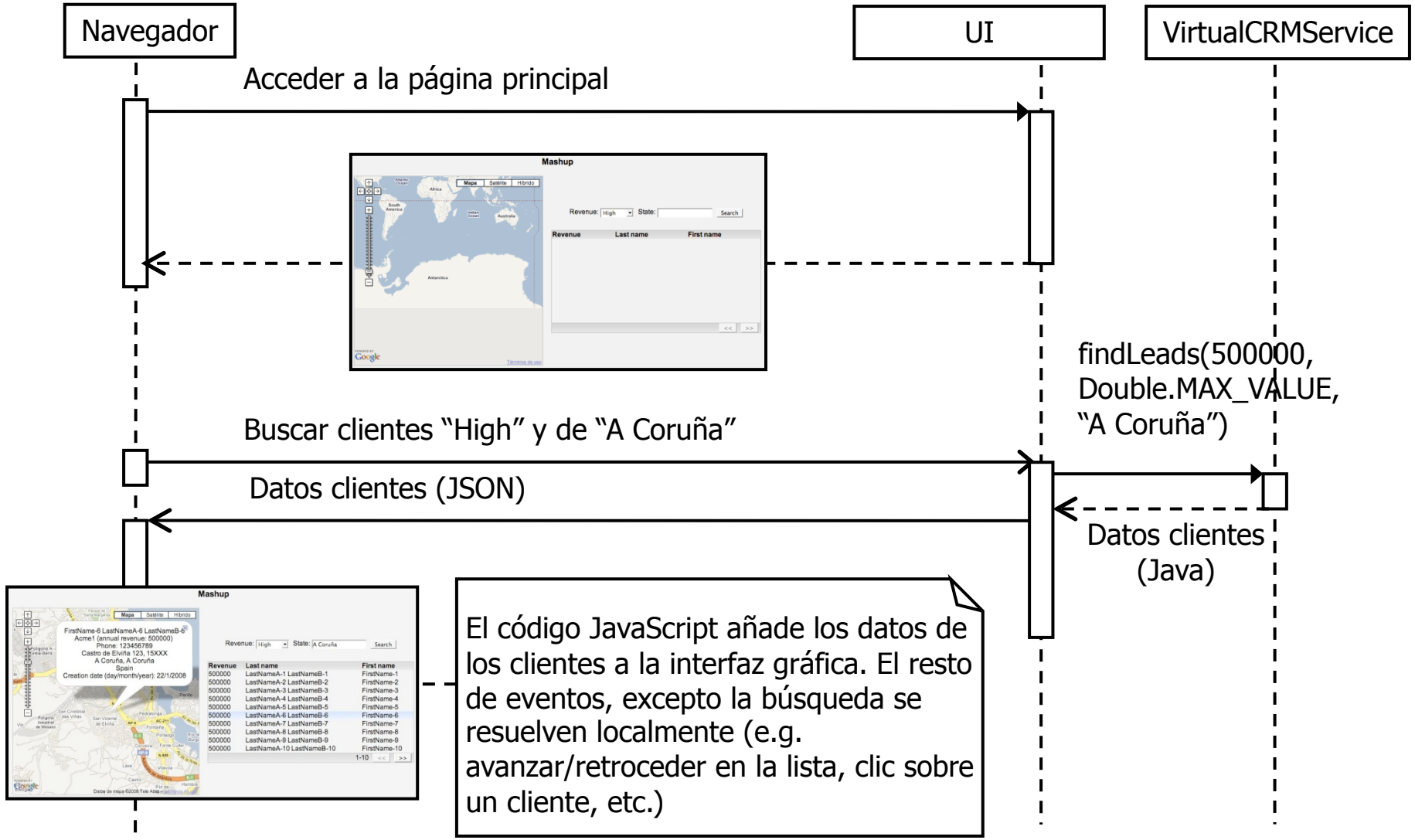
Práctica: arquitectura (1)



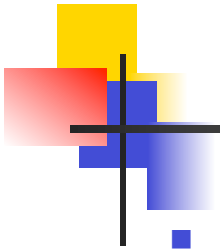
Práctica: arquitectura (2)



Práctica: arquitectura (3)



Práctica: arquitectura (y 4)



- Se proporciona código de partida, estructurado en dos módulos
  - **ui**
    - Interfaz 100% AJAX implementada con GWT
  - **virtualcrm**
    - Interfaz `VirtualCRMService`
      - Existe una única instancia (Singleton) que se puede obtener mediante `VirtualCRMServiceFactory.getVirtualCRMService()`
      - La clase de implementación puede tener estado global no modificable
      - **Se deberá implementar con una arquitectura por capas de acceso a servicios**
      - **Acceso a Salesforce: opcional para Master**
- Se deberán crear dos módulos adicionales
  - **leadnews**
    - Servicio RSS con información de clientes recientes
  - **internalcrm**
    - Representa el servicio de búsqueda del CRM interno
    - **Se deberá realizar una sencilla implementación ficticia**
- Además
  - **Uso de Eclipse BPEL Designer y ActiveBPEL para implementar un flujo**
    - **Definición del flujo (opcional para Master)**
    - **Parte opcional para Ingeniería: implementación del flujo (requiere añadir más módulos)**