# Presentación técnica

## Objetivos

* Integración de información proveniente de diversas fuentes
* Control de acceso
* Acceso remoto a datos almacenados en la base de datos a través de consultas
* Análisis de datos sobre conductores
* Automatización de procesos de actualización y recuperación de la base de datos
* Administración de cuentas de usuarios (clientes, administradores y operadores)
* [Usabilidad]

## Objetivos de diseño

* Minimización de acoplamiento
* Reutilización de componentes
* Facilidad de extensión de funcionalidad
* …

## Frameworks y tecnologías

(logo de cada cosa)

* Java
* Wicket
* JXL
* Tomcat
* Maven
* Hibernate
* MySQL
* Eclipse
* Log4j
* SVN

## Componentes

## Despliegue

## 1.Casos de Usos

Actor: Cliente

**Acceder a cuenta de usuarios:** caso de uso que permite al cliente acceder a su cuenta de usuario.

(Completar)

Actor: Administrador

**Mantener cuenta de usuarios:**

Actor: Operador

**Actualizar la BD:** caso de uso que permite mantener la base de datos del sistema actualizada.

2.Arquitectura(Completar)

Presentación

* **Wicket Apache**: es un proyecto de [código abierto](http://es.wikipedia.org/wiki/C%C3%B3digo_abierto), disponible bajo los términos de la [licencia Apache](http://es.wikipedia.org/wiki/Licencia_Apache). Hace énfasis a uso de "puro [Java](http://es.wikipedia.org/wiki/Java_%28lenguaje_de_programaci%C3%B3n%29)" y "puro [HTML](http://es.wikipedia.org/wiki/HTML)", a diferencia de tecnologías como [JSP](http://es.wikipedia.org/wiki/JavaServer_Pages) o [JSF](http://es.wikipedia.org/wiki/JavaServer_Faces) que permiten incluir lógica de aplicación en las plantillas de presentación.
* **Componentes, Html**:

Lógica del Negocio:

* **Clases de dominio en Java:** son aquellas que realizan la lógica de la aplicación, como es el control de permisos,
* **Transacciones y Servicios:**
* **Actualización de la base de datos**: aquí se encuentran todos los procesos que actualizan la base de dato, ya sea en forma interactiva desde la pagina como son los procesos que se corren para incorporar las actualizaciones en la base de datos.

Persistencia:

* **Hibernate**: utilizamos hibernate que es un motor de persistencia de código abierto, básicamente lo que hace es una traducción entre registro y objeto.
* **JDBC**: es un API para trabajar con bases de datos desde Java, independientemente de la base de datos a la que accedemos. A la hora de conectarnos a una base de datos usando JDBC usamos un driver intermedio, en nuestro caso el driver para trabajar con la base de datos MySQL, es com.mysql.jdbc.Driver.
* **Soporte de lenguaje de consultas**: utilizamos como lenguaje de consulta SQL, dado que resulta muy simple poder acceder, actualizar o insertar datos a las tablas mediante este lenguaje.
* **Base de Datos**: Utilizamos MySQL.

**Ventajas**

* + MySQL software es Open Source
  + Velocidad al realizar las operaciones, lo que le hace uno de los gestores con mejor rendimiento.
  + Soporta gran variedad de Sistemas Operativos
  + Baja probabilidad de corromper datos, incluso si los errores no se producen en el propio gestor, sino en el sistema en el que está.
  + Su conectividad, velocidad, y seguridad hacen de MySQL Server altamente apropiado para acceder bases de datos en Internet

**Desventajas**

* + Un gran porcentaje de las utilidades de MySQL no están documentadas.
  + No es intuitivo, como otros programas (ACCESS).

### Ventajas

* Multiplataforma
* Actualización dinámica de la aplicación (el sitio web al que accede el cliente siempre está actualizado)
* Independencia del sistema operativo
* Independencia de la base de datos
* No se requiere instalación
* No es necesario aplicar seguridad en los equipos de clientes

## Modelo de datos

[diagrama]

Aca se puede explicar algo del formato de los archivos de actualización.

## Interfaces con otros sistemas

[Diagrama]

* Descripción de protocolos. Se puede explicar uno de los del doc como ejemplo.

## Seguridad

* URL encriptado
* Control de acceso a través de sesiones de usuario.
* Definición de perfiles que determinan qué provilegios tiene cada tipo de usario.
* Validación de campos
* [Seguridad en la base de datos, MD5, DES, TRIPLE DES algo así??? Y GRANTS]
* [ Si logro agregarlo] Completely Automated Public Turing test to tell Computers and Humans Apart (Captcha)

La implementación de captchas, URLs encriptadas y la validación de campos permite evitar:

* SQL injection. [No es posible especificar como parámetros consultas SQL en el URL o en campos.]
* Cross Site Scripting (XSS). [No es posible ejecutar código a través de campos o URLs.]
* Spam en el formulario de contratación de servicios.

## Mejoras

* Implementación de estadísticas para conocer características de los clientes.