# 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

* Java: lenguaje de programación orientado a objetos, que puede correr en distintas plataformas. El proyecto fue desarrollado completamente en java.
* Wicket: framework java para desarrollo web. Permite desarrollar la vista del sitio web, incorporando código javascript y AJAX facilemente. A partir de este framework, fue posible el diseño de formularios e interfaces en general para poder acceder al sistema y modificar la base de datos.
* JXL: biblioteca en java que permite leer y modificar spreadsheets de excel. La necesidad de la biblioteca surge al momento de actualizar la base de datos dado que se utilizan archivos en excel.
* Tomcat: [servidor web](http://es.wikipedia.org/wiki/Servidor_web) que funciona como un contenedor de [servlets](http://es.wikipedia.org/wiki/Servlets) y JSPs. Permite probar nuestro proyecto en el localhost.
* Maven: permite organizar y administrar un proyecto. Entre otras opciones, permite la incorporación de dependencias y modificación del classpath del proyecto a través de un archivo xml de especificaciones. En el proyecto fue necesario utilizar este framework para lograr la integración de los frameworks y tecnologías involucrados.
* Hibernate: framework de persistencia que permite realizar el mapeo del modelo relacional al modelo de objetos y viceversa. Es una herramienta de [Mapeo objeto-relacional](http://es.wikipedia.org/wiki/Mapeo_objeto-relacional) (ORM) que permite acceder y modificar la base de datos en el código java del proyecto.
* MySQL: sistema de gestión de bases de datos, que permite, entre otras cosas: crear bases de datos, crear tablas, realizar consultas. Se decidió utilizar esta tecnología en el proyecto porque es muy fácil su integración con aplicaciones web.
* Eclipse: entorno de desarrollo integrado y multiplataforma. Para el proyecto utilizamos la versión disponible para J2EE. Permite detectar errores, correr la aplicación en el servidor, etc.
* Toad: [aplicación informática](http://es.wikipedia.org/wiki/Aplicaci%C3%B3n_inform%C3%A1tica) de desarrollo [SQL](http://es.wikipedia.org/wiki/SQL) y administración de [base de datos](http://es.wikipedia.org/wiki/Base_de_datos). Se utilizó en el proyecto para desarrollas consultas sql a través de un entorno más amigable.
* SVN: [sistema de control de versiones](http://es.wikipedia.org/wiki/Revision_Control_System). Se configuró un repositorio de forma tal que los miembros del grupo pudieran acceder y modificar el mismo conjunto de datos del proyecto. De esta forma, cada integrante del grupo accede a la versión más actualizada del proyecto y se simplifica la colaboración en el desarrollo del mismo.

## Componentes

En este diagrama podemos observar las dependencias entre los distintos componentes del sistema.

Inicialmente, un usuario del sistema utilizará la interfaz de usuario (componente web) para poder acceder al sistema.

El manejo de sesiones, cuentas de usuario, autenticación se encuentra contenido en el componente de **Sesiones**. Asimismo, este componente depende del componente de **Seguridad**, dado que es necesario determinar qué permisos tienen un usuario en relación al sistema. Esta funcionalidad se encuentra en el componente de Seguridad, junto con procesos relacionados con el control de acceso y modificación de la base de datos.

Además, se desarrolló un componente dedicado a la administración de información relacionada con los **clientes**. Este componente permite determinar los servicios, las consultas máximas disponibles para cada uno de los servicios, etc.

Existe un componente de donde se procesan las consultas realizadas por el usuario y se obtienen los resultados necesarios para la generación del informe. Este componente el de **Consultas**.

El tratamiento de los resultados de una consulta, así como la presentación de los mismos se encuentra desarrollado en el componente de **Informes**.

Finalmente, se desarrollaron componentes relacionados con los procesos de actualización y recuperación de la base de datos. Estos componentes incluyen procesos que acceden a la base de datos a través de JDBC.

## 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.

**Consulta datos relacionados con personas**

(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. Se realiza a través de un archivo Excel en el cual el operador mediante la interfaz web sube el archivo para la actualización de datos.

2. Arquitectura(Completar)

Capa de 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.