
UOC
Universitat Oberta
de Catalunya

AlcoNQL

Herramienta de consulta SQL por medio de lenguaje Natural

Diego Alconada González

2º Ciclo Ingeniería Informática

Jordi Duran Cals

16/01/2013

Control del Cambios

Histórico de Versiones		
Versión	**Fecha**	**Resumen de los cambios**
1.0	27/09/2012	Creación del Formato Documento.
1,1	30/09/2012	Introducción, objetivos y planificación
1.2	03/10/2012	Tareas, Diagrama de Gantt
2.1	11/10/2012	Estudio procesamiento lenguaje natural
2.2	20/10/2012	Estudio sentencias SQL
2.3	28/10/2012	Estudio herramientas interpretación lenguaje natural
2.4	02/11/2012	Definición de entorno
2.5	07/11/2012	Definición de subconjunto de lenguaje natural
3.1	20/11/2012	Interpretación de consultas SQL básicas
3.2	25/11/2012	Implementación de clausula WHERE sencilla
3.3	7/12/2012	Aumento de complejidad en clausula WHERE e implementación de cláusulas BETWEEN e IN
3.4	10/12/2012	Creación de entorno grafico e instalador
3.5	12/12/2012	Redacción de la memoria

Contenido

1 INTRODUCCIÓN

1.1 Contexto y justificación del Proyecto

Como punto final a la carrera, el proyecto sirve para poner en práctica todos los conocimientos y técnicas adquiridos. Así como el análisis, planificación y desarrollo del mismo. Con el proyecto fin de carrera se da el último paso en la formación del alumno.

Con este proyecto en particular se profundizara en las consultas de bases de datos, compiladores para poder interactuar con el lenguaje natural

Con el paso del tiempo la tecnología se ha ido haciendo accesible a todo el público y ha crecido el interés de interactuar con ella. Pero muchas veces se requieren conocimientos técnicos.

Para poder eliminar estas barreras y acercar la tecnología a todo el público ha aparecido la necesidad de crear herramientas que permitan interactuar con las interfaces de las aplicaciones por medio del lenguaje natural.

Por ello, con este proyecto se pretende hacer una introducción a la creación y uso de herramientas que interpreten el lenguaje natural.

1.2 Motivaciones

Con este proyecto se pretende realizar una herramienta que sea capaz de interpretar sentencias en lenguaje natural y convertirlas en consultas SQL

Como motivación personal me permitirá comprobar cuales han sido mis conocimientos adquiridos durante la carrera y demostrarme si soy capaz de realizar un proyecto de inicio a fin.

1.3 Objetivos del proyecto

- Estudio del procesamiento del lenguaje natural y acotado de subconjunto del lenguaje natural para ser usado en el proyecto
- Estudio de la sintaxis de consultas SQL (SELECT) y delimitado del uso de las mismas
- Estudio de las herramientas ya desarrolladas relacionadas con el objetivo del proyecto

- Desarrollo de una herramienta que interprete el lenguaje natural convirtiendo sentencias en leguaje natural en consultas SQL, realice estas consultas en una base de datos y muestre los resultados.

1.4 Estructura de la memoria del proyecto

La memoria del proyecto se estructura en los siguientes capítulos

1. **Introducción**: En el primer capítulo se detallaran el contexto y justificación del proyecto las motivaciones y los objetivos del proyecto.
2. **Estudio de la viabilidad**: Se realiza un estudio del procesamiento del lenguaje natural, así como de herramientas de análisis que puedan facilitar el trabajo. Se identificaran las estructuras y sentencias SQL que vayan a ser incluidas en el proyecto y herramientas de interpretación del lenguaje natural para comprender cuales son las posibles soluciones ya existentes. Se definirá el subconjunto de lenguaje natural utilizado. Por último se realizara una estimación de costes y beneficios
3. **Diseño técnico e implementación**: Comienza con el modelo de desarrollo seguido para continuar con la definición del entorno usado para el desarrollo. También se definen los diagramas de diseño como casos de uso, diagramas de clases y diagramas de secuencia. Para finalizar con la descripción de la implementación de la herramienta, dando detalles de los distintos componentes como los analizadores léxico y sintáctico, el control de errores y la definición de la interfaz gráfica de usuario
4. Test y pruebas: Se describen las pruebas realizadas contra la base de datos y sobre la herramienta gráfica. También se detallan ejemplos de uso que también sirvieron como test
5. Líneas futuras y Conclusiones
6. Bibliografía
7. Anexo

2 ESTUDIO DE LA VIABILIDAD

En este apartado se detallarán las tareas a realizar y se marcará una planificación para las mismas, se realizará un estudio de la teoría del procesamiento del lenguaje natural, haciendo hincapié en el estudio de la morfología y clases de palabras y el estudio de la sintaxis gramatical. También se estudiaran y valorará el uso de herramientas de análisis del lenguaje natural, estudio de sentencias SQL y herramientas de interpretación del lenguaje natural con el fin de poder definir los pasos a seguir para desarrollar el producto. También se definirá el subconjunto del lenguaje natural que será utilizado. Por último se realizara una estimación de costes y beneficios

2.1 Descripción

Este proyecto trata de procesar sentencias en lenguaje natural transformándolas en consultas SQL SELECT y poder realizar la consulta a una base de datos sin tener que conocer el lenguaje técnico que se precisa para ello.

El proyecto se puede disgregar en hitos más sencillos:

- Realización de la herramienta de transformación del lenguaje natural en consultas SQL (Parte principal del proyecto)
- Creación de la base de datos con la que interactuar
- Establecer la comunicación entre la herramienta de transformación del lenguaje natural en consultas SQL con la base de datos
- Creación de una interfaz gráfica de usuario que agrupe todas las funcionalidades

2.2 Tareas

2.2.1 Generales

- G0 – Planificación del proyecto
- G1 – Redacción del resumen
- G2 – Redacción de la introducción
- G3 – Redacción de los capítulos
- G4 – Glosario
- G5 – Bibliografía

2.2.2 Especificas

PAC2

- Estudio del procesamiento del lenguaje natural
- Estudio de herramientas de consulta SQL por medio del lenguaje natural
- Estudio de sentencias SQL y definición de ámbito de desarrollo
- Definición del subconjunto del lenguaje natural a usar
- Definición del entorno de desarrollo

PAC3

- Desarrollo herramienta requisitos mínimos
- Implementación control de errores
- Aumento de la funcionalidad de la herramienta

2.2.3 Descripción de las tareas especificas

- **Estudio del procesamiento del lenguaje natural**: Estudio de las técnicas y herramientas para interpretar el lenguaje natural. Las dificultades que conlleva y los beneficios que puede aportar.
- **Estudio de herramientas de consulta SQL por medio del lenguaje natural**: Estudio de herramientas como SQ-HAL. Sus ventajas e inconvenientes y técnicas que se han usado en su desarrollo.
- **Estudio de sentencias SQL y definición del ámbito de desarrollo**: Estudio de las sentencias SQL, en especial SELECT y evaluación de su complejidad. Estudio de las llamadas recursivas. Identificación de las sentencias INSERT y DELETE para una posible implementación.
- **Definición del subconjunto de lenguaje natural a usar**: Definición de subconjuntos de lenguaje natural. Partiendo del más restrictivo y cercano a la estructura de las sentencias SQL para terminar usando el más cercano al lenguaje natural. Se definirá un lenguaje restrictivo que posteriormente se ira enriqueciendo.
- **Definición del entorno de desarrollo**: Preparación del entorno con las herramientas para desarrollar (JLEX. CUP, MySQL).
- **Desarrollo de los requisitos mínimos de la herramienta:** Desarrollo del analizador léxico, sintáctico y semántico que sean capaces de ejecutar consultas SQL básicas con lenguaje natural

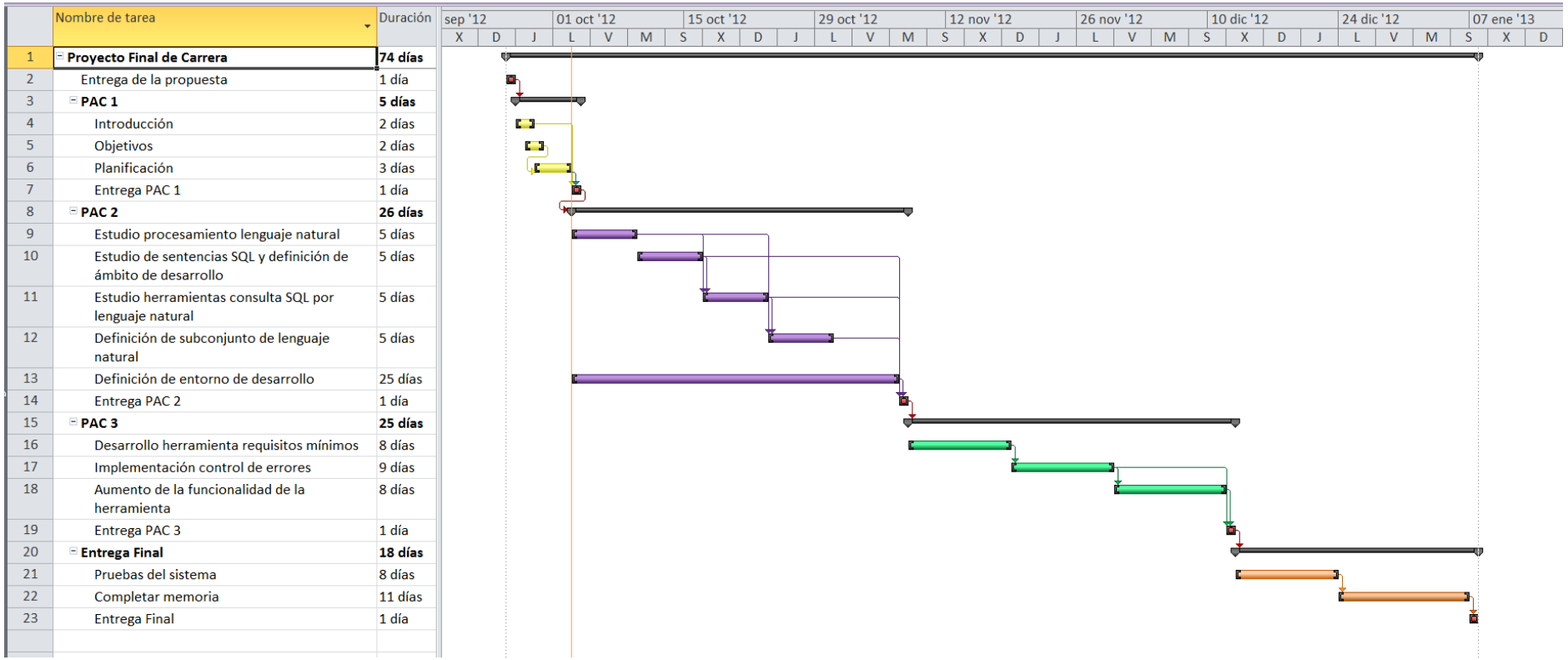
- **Implementación de control de errores**: Implementación del control de errores de la herramienta desarrollada en el punto anterior
- **Aumento de la funcionalidad de la herramienta**: Se implementaran consultas SELECT más complejas (como sentencias SQL) y se aumentará el subconjunto de lenguaje natural

2.3 Planificación

2.3.1 Tareas

Nombre de tarea	Duración	Comienzo	Fin
Proyecto Final de Carrera	**74 días**	**mié 26/09/12**	**lun 07/01/13**
Entrega de la propuesta	1 día	mié 26/09/12	mié 26/09/12
PAC 1	**5 días**	**jue 27/09/12**	**mié 03/10/12**
Introducción	2 días	jue 27/09/12	vie 28/09/12
Objetivos	2 días	vie 28/09/12	sáb 29/09/12
Planificación	3 días	sáb 29/09/12	mar 02/10/12
Entrega PAC 1	1 día	mié 03/10/12	mié 03/10/12
PAC 2	**26 días**	**mié 03/10/12**	**mié 07/11/12**
Estudio procesamiento lenguaje natural	5 días	mié 03/10/12	mar 09/10/12
Estudio de sentencias SQL y definición de ámbito de desarrollo	5 días	mié 10/10/12	mar 16/10/12
Estudio herramientas consulta SQL por lenguaje natural	5 días	mié 17/10/12	mar 23/10/12
Definición de subconjunto de lenguaje natural	5 días	mié 24/10/12	mar 30/10/12
Definición de entorno de desarrollo	25 días	mié 03/10/12	mar 06/11/12
Entrega PAC 2	1 día	mié 07/11/12	mié 07/11/12
PAC 3	**25 días**	**jue 08/11/12**	**mié 12/12/12**
Desarrollo herramienta requisitos mínimos	8 días	jue 08/11/12	dom 18/11/12
Implementación control de errores	9 días	lun 19/11/12	jue 29/11/12
Aumento de la funcionalidad de la herramienta	8 días	vie 30/11/12	mar 11/12/12
Entrega PAC 3	1 día	mié 12/12/12	mié 12/12/12
Entrega Final	**18 días**	**jue 13/12/12**	**lun 07/01/13**
Pruebas del sistema	8 días	jue 13/12/12	dom 23/12/12
Completar memoria	11 días	lun 24/12/12	dom 06/01/13
Entrega Final	1 día	lun 07/01/13	lun 07/01/13

2.3.2 Diagrama de Gantt



	Nombre de tarea	Duración
1	**Proyecto Final de Carrera**	**74 días**
2	Entrega de la propuesta	1 día
3	**PAC 1**	**5 días**
4	Introducción	2 días
5	Objetivos	2 días
6	Planificación	3 días
7	Entrega PAC 1	1 día
8	**PAC 2**	**26 días**
9	Estudio procesamiento lenguaje natural	5 días
10	Estudio de sentencias SQL y definición de ámbito de desarrollo	5 días
11	Estudio herramientas consulta SQL por lenguaje natural	5 días
12	Definición de subconjunto de lenguaje natural	5 días
13	Definición de entorno de desarrollo	25 días
14	Entrega PAC 2	1 día
15	**PAC 3**	**25 días**
16	Desarrollo herramienta requisitos mínimos	8 días
17	Implementación control de errores	9 días
18	Aumento de la funcionalidad de la herramienta	8 días
19	Entrega PAC 3	1 día
20	**Entrega Final**	**18 días**
21	Pruebas del sistema	8 días
22	Completar memoria	11 días
23	Entrega Final	1 día

2.4 Teoría del procesamiento del lenguaje natural

2.4.1 Introducción

Rama de la inteligencia artificial encargada de crear herramientas para poder interpretar el lenguaje. Desde la aparición de los ordenadores se ha estudiado esta disciplina, para facilitar la interactuación entre hombre y máquina.

La interpretación del lenguaje no es tarea sencilla. La interpretación de las palabras, con sus distintas variaciones (prefijos, sufijos, plurales, diminutivos, etc.). Unido a los distintos significados que pueden adquirir las mismas.

Una vez identificada la problemática de las palabras hay que profundizar en la estructura de las frases, o la sintaxis. La gramática es muy variada por lo que hay que identificar que papel juega cada palabra en una frase. Identificar cual el verbo que da la acción, o el sujeto que la desempeña.

Y por último la semántica. Interpretar el significado de una frase puede ser tarea ardua. Puede haber distintas frases con un mismo significado o Frases que cambien completamente su significado con cambios sutiles en su estructura.

2.4.2 Estudio de las palabras

Expresiones regulares

Formalmente, una expresión regular es una notación algebraica para caracterizar un conjunto de símbolos. Con las expresiones regulares podemos definir patrones que buscar en textos.

A continuación se muestran las reglas para definir expresiones regulares POSIX:

Expresión	Descripción
.	Cualquier carácter
\char	Caracteres no alfanuméricos
\n	Fin de línea
\r	Retorno de carro
\t	Tabulador
[…]	Cualquier carácter dentro de los corchetes
[…-…]	Cualquier carácter del rango
[^…]	Cualquier carácter q no esté en el corchete
[^…-…]	Cualquier carácter que no esté en el rango
^	Comienzo de línea
$	Fin de línea
\b	Límite de la palabra
\B	Indica que no termina la palabra
*	Cero o más repeticiones

+	Una o más repeticiones
?	Cero o 1 repetición del carácter precedente
{n}	n repeticiones del carácter precedente
{n,m}	De n a m repeticiones del carácter precedente
{n, }	Al menos n repeticiones del carácter precedente
.*	Cualquier conjunto de caracteres
(…)	Agrupación para preferencia
…|…	Una u otra
\d	Cualquier digito
\D	Cualquier no digito
\w	Cualquier alfanumérico
\W	Cualquier no alfanumérico
\s	Espacio
\S	Cualquier carácter menos el espacio

Análisis morfológico

Es necesario conocer las reglas ortográficas y morfológicas para poder identificar los plurales de las palabras, las conjugaciones de los verbos, los diminutivos, etc.

Definir una amplia gama de reglas ayudara a producir un diccionario del lenguaje más completo y robusto.

El estudio de la morfología de las palabras implica identificar la unidad mínima con significado de cada una de ellas. Identificar la raíz y los afijos.

Por ejemplo la palabra “perros”, consta de la raíz “perro” y el afijo (sufijo) “s”

Hay muchas formas de combinar morfemas para crear palabras. Algunos de los procesos más importantes son: inflexión, derivación, composición

- **Inflexión**: alteración de la palabra por medio de morfemas para expresar sus distintas funciones (conjugaciones, declinaciones, etc.)
- **Derivación**: formación de palabras a partir de otras añadiendo afijos
- **Composición**: creación de nuevas palabras por medio de la conjugación de varios lexemas

Clases de palabras

El estudio de las clases de palabras nos ayudara posteriormente en la sintaxis gramatical. El poder englobarlas en distintos grupos, nos ayudará a poder predecir las palabras siguientes o poder definir el significado de la palabra por su contexto.

- **Nombre**: el sujeto de la frase. Da “nombre” a lugares, personas, o cosas

- **Verbo**: encargado de dar la acción a la frase. Las conjugaciones de los verbos pueden ser complicadas de interpretar
- **Adjetivos**: describen propiedades o cualidades
- **Adverbios**: modificador de un verbo, adjetivo u otro adverbio
- **Preposiciones**: encargadas de crear las uniones en una oración.
- **Conjunciones**: nexos entre frases
- **Pronombres**: señalan o representan personas objetos o hechos ya conocidos.

2.4.3 Estudio de la sintaxis gramatical

El conocimiento de la sintaxis gramatical es necesario para el estudio del lenguaje natural.

Básicamente una frase está constituida por un sintagma nominal y un sintagma verbal. El sintagma nominal es un grupo de palabras cuyo núcleo es un nombre. El sintagma verbal está encabezado por un verbo no auxiliar.

También tenemos sintagmas preposicionales, adjetivales y adverbiales.

Gramáticas de contexto libre

Es uno de los sistemas matemáticos más usado para la creación de estructuras gramaticales. Consiste en crear un conjunto de reglas que defina como se pueden agrupar los símbolos del lenguaje.

Constan de dos clases de símbolos:

- **Terminales**: corresponden a palabras del lenguaje
- **No terminales**: se corresponden a uniones de símbolos terminales y no terminales

La estructura de una regla de contexto libre es la siguiente:

Símbolo no terminal → Símbolo no terminal | Símbolo terminal

Pudiendo combinarse los símbolos de la parte derecha tantas veces como se quiera.

Todas las gramáticas tienen asociado un árbol de análisis. Donde los símbolos terminales estarán situados en la parte más baja del mismo. Estos se irán agrupando en símbolos no terminales, que a su vez se irán agrupando en más símbolos no terminales, terminando en el símbolo no terminal de inicio. Toda gramática de contexto libre tiene que tener un símbolo de comienzo.

Ejemplo:

Símbolos terminales:

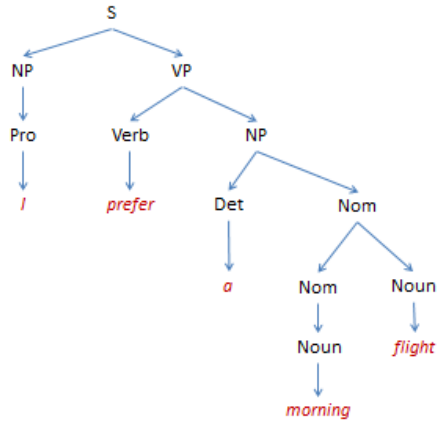
- *Noun* → *flights | breeze | trip | mornings*
- *Verb* → *is | prefer | like | need | want | fly*
- *Adjetive* → *cheapest | non-stop | first | latest*
- *Pronoun* → *me | I | you | it*
- *Proper-Noun* → *Alaska | Baltimore | Los Angeles*
- *Determiner* → *the | a | an | this | these | that*
- *Preposiocion* → *from | to | on | near*
- *Conjunction* → *and | or | but*

Reglas:

- *S* → *NP VP*
- *NP* → *Pronoun | Proper-Noun | Determiner Nominal | Noun*
- *Nominal* → *Nominal Noun | Noun*
- *VP* → *Verv | Verb NP | Verb NP PP | Verb PP*

Símbolo no terminal de inicio →S

Árbol



Analizadores sintácticos

El analizador sintáctico es el encargado de reconocer una sentencia y asignarle una estructura sintáctica. Basándose en las reglas definidas en la gramática que se le haya proporcionado, buscara la estructura que mejor se ajuste a las reglas.

Hay dos estrategias para realizar esta operación. Análisis de arriba abajo y análisis de abajo a arriba.

Diferentes técnicas de análisis

- **Análisis de arriba abajo**: Intenta crear un árbol intentando construirlo desde el símbolo de inicio hasta llegar a los símbolos terminales. El árbol se da por completado cuando se consigue relacionar todas las palabras de entrada con símbolos terminales
- **Análisis de abajo a arriba**: El análisis se inicia por las palabras que se dan como entrada. Buscando sus símbolos terminales e intentando crear el árbol a partir de ellos. El árbol se da por completado cuando se consigue llegar al símbolo de inicio.

Cada uno tiene sus ventajas y diferencias. El análisis de arriba a abajo nunca perderá el tiempo explorando árboles que no vayan a dar como resultado el símbolo inicial (ya que siempre se parte de este estado). Por el contrario el análisis de abajo a arriba crear arboles a partir del estado inicial que posteriormente no podrán ser terminados por no poder ajustarse a ninguna regla, por lo que esa opción tendrá que ser abandonada

Ambigüedad estructural

La ambigüedad es uno de los mayores problemas de los analizadores sintácticos. Esta sucede cuando la gramática ofrece más de una posibilidad para definir una sentencia. Con lo que el analizador no sabrá discernir de cuál es la buena.

Analizadores estadísticos

Los análisis estadísticos pretenden solucionar el problema de las ambigüedades a la hora de analizar una sentencia y la interpretación de los modelos de lenguaje, ya que estos no siempre se rigen por reglas estrictas.

El sistema más utilizado es la gramática libre de contexto probabilístico. Este sistema es una mejora de las gramáticas de contexto libre. Asignando una probabilidad a cada regla que representa la probabilidad:

$A \rightarrow B\ [p]$ *siendo p la probabilidad de que el símbolo A se convierta en B*

Con ello se pueden definir distintas reglas para el mismo símbolo, asociándoles una probabilidad de ejecución.

2.4.4 FreeLing

Librería orientada al desarrollo que provee servicios para el análisis del lenguaje, siendo capaz de interpretar lenguajes como: australiano, catalán, inglés, gallego, italiano, portugués, ruso, español y gales.

Con esta herramienta se puede realizar un estudio pormenorizado de una sentencia. Pudiendo interpretar así sus partes léxicas y gramaticales. Este estudio ayuda a una posterior interpretación de la sentencia.

Herramienta muy potente para realizar el análisis de una sentencia, pero de uso complejo. No ha sido utilizada en el desarrollo del producto por requerir un elevado coste de tiempo en comprender y conocer el correcto uso de la misma.

Formado por módulos que cubren análisis básicos de lenguaje, con la combinación de ello se puede llegar a hacer un análisis profundo y preciso. A continuación se procede a dar una descripción de los modulos.

Identificador de lenguaje

Este no es un módulo de análisis propiamente dicho, ya que no enriquece el texto con información. Este módulo compara el texto proporcionado con los modelos disponibles de los diferentes lenguajes. Devuelve un Array de probabilidades identificando a que lenguaje puede pertenecer.

Módulo "Tokenizer"

Convierte el texto plano en un vector de conjunto de caracteres acorde a las reglas de definidas para la tokenizacion del lenguaje.

Este es el primer módulo por el que tiene que pasar un texto para ser analizado. Ya que este módulo lo que hace es reconocer las expresiones mínimas con significado (lo que podríamos reconocer como palabras) "Tokens" por medio de expresiones regulares.

Módulo "Splitter"

Realiza agrupaciones de tokens recibidos del módulo "Tokenizer" que finalizan al recibir un token de final de frase. Formando así una agrupación de sentencias

Módulo analizador morfológico

Modulo manejador de submódulos que interactúan entre si para obtener toda la información de las sentencias.

Estos submódulos facilitan la identificación de números, fechas, ratios, porcentajes, magnitudes físicas, unidades monetarias, signos de puntuación, etc. Asigna las categorías gramaticales a las palabras que formas una sentencia. Busca en un diccionario las palabras para identificar su lexema, reconoce grupos de palabras con significados explícitos. Revisa la ortografía de las palabras, para que estas puedan ser identificadas correctamente. Identifica los significados de las palabras. Resuelve ambigüedades. Realiza un gráfico en forma de árbol del análisis realizado, etc.

En definitiva realiza un estudio del texto proporcionado enriqueciéndolo con información que facilita el análisis e interpretación del mismo.

2.5 Estudio de sentencias SQL

2.5.1 Estructura SQL SELECT básica

La estructura básica de la sentencia SELECT es:

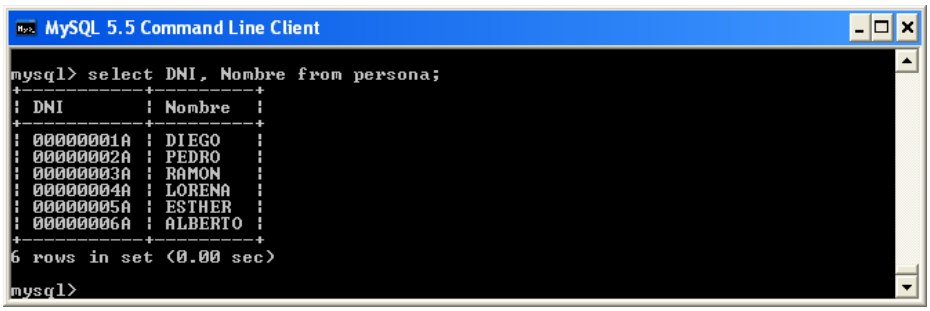
*SELECT * FROM TABLENAME*

Todas las sentencias SELECT tienen que empezar con la palabra SELECT.

Seguido de los campos que se quieren mostrar, en este caso con * indicamos que nos muestre todo.

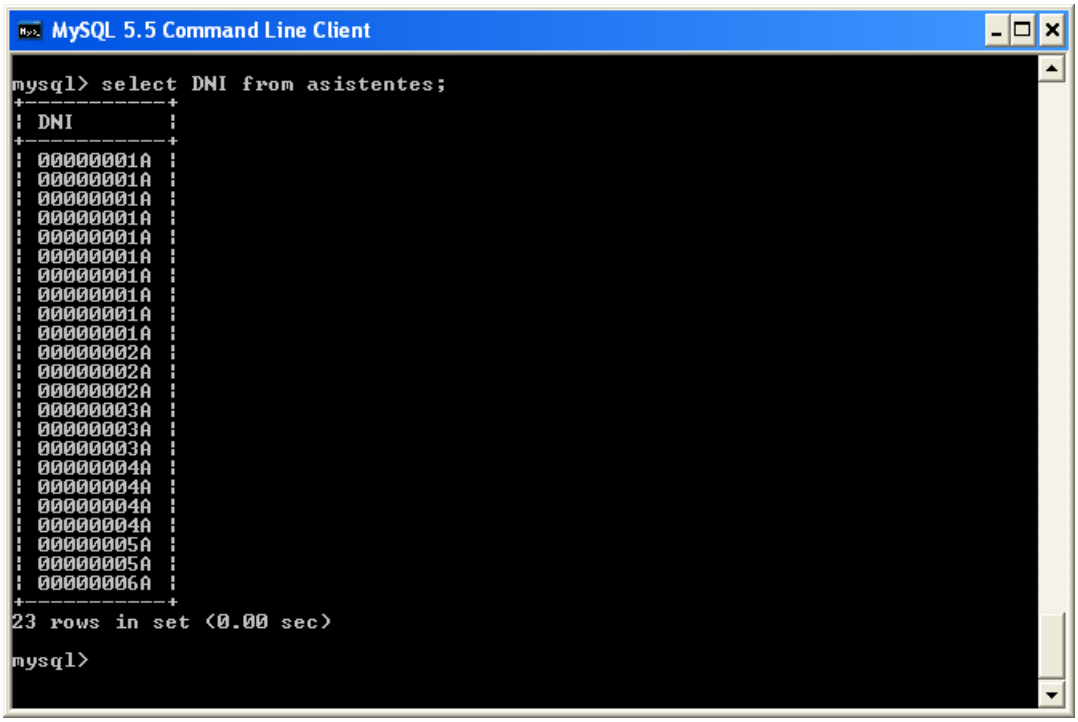
Por ultimo hay que indicar la tabla sobre la que se realiza la SELECT.

Si se quiere mostrar solo alguno de los campos se pueden indicar después de la palabra SELECT, los ID´s de los atributos separados por una coma (‘,’)

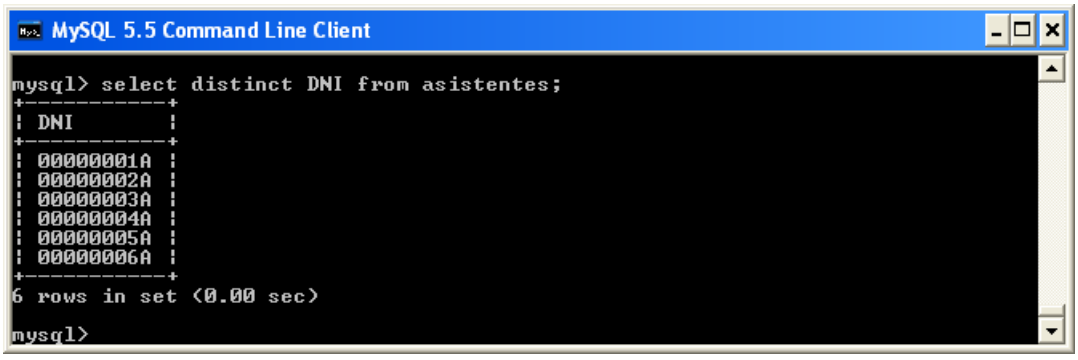


Un atributo puede tener registros repetidos. Si se quiere mostrar solo los valores distintos, y no los valores repetidos se tienen que usar la palabra DISTINCT después de la palabra SELECT

Atributo con registros repetidos



Selección solo de los distintos



2.5.2 Clausula WHERE

La cláusula WHERE se puede añadir al final para que se muestren solo los registros que cumplen la restricción que se indica después del WHERE. La restricción se indicará dando un valor específico de uno o varios atributos de la tabla

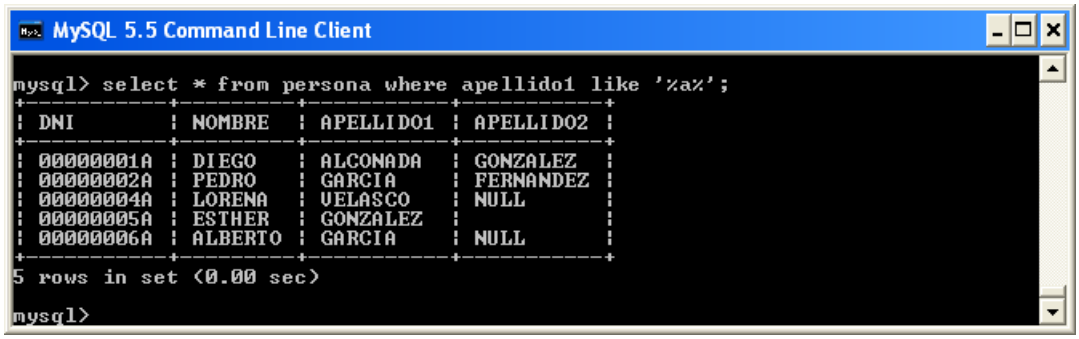
Si se está especificando un valor concreto de un atributo, hay que tener en cuenta que si el atributo es de tipo VARCHAR hay que indicar su valor entre comillas simples (‘…’). Si se trata de un NUMBER se pone directamente

Los operadores permitidos para la cláusula WHERE son:

Operador	Descripción
=	Igual
<> (o !=)	Distinto
>	Mayor que
<	Menor que
>=	Mayor o igual que
<=	Menor o igual que
BETWEEN	Incluido en un rango
LIKE	Como un patrón
IN	Que se encuentra en un conjunto

Los operadores AND y OR sirven para unir condiciones en la cláusula WHERE

Con el operador LIKE se puede buscar un patrón en un atributo



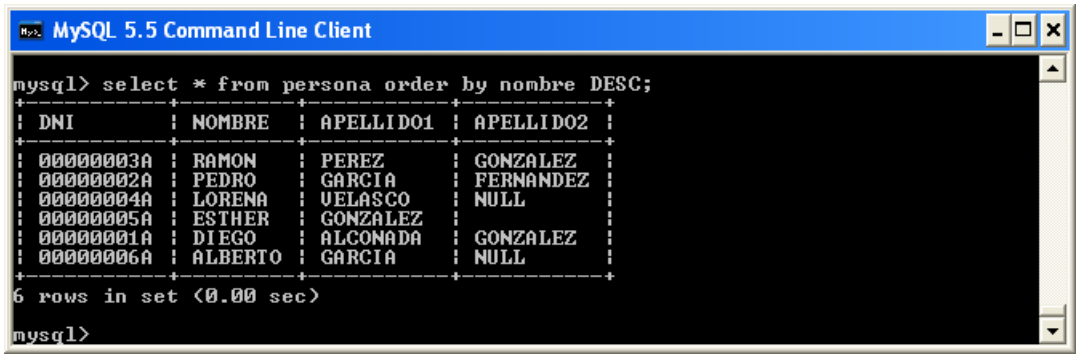
Con la cláusula IN se pueden indicar múltiples valores

Con la cláusula BETWEEN se selecciona un rango de valores

2.5.3 ORDER BY

Con ORDER BY se pueden ordenar los registros por un atributo especificado

Se puede indicar que se ordene de forma descendente añadiendo al final DESC



2.5.4 LIMIT

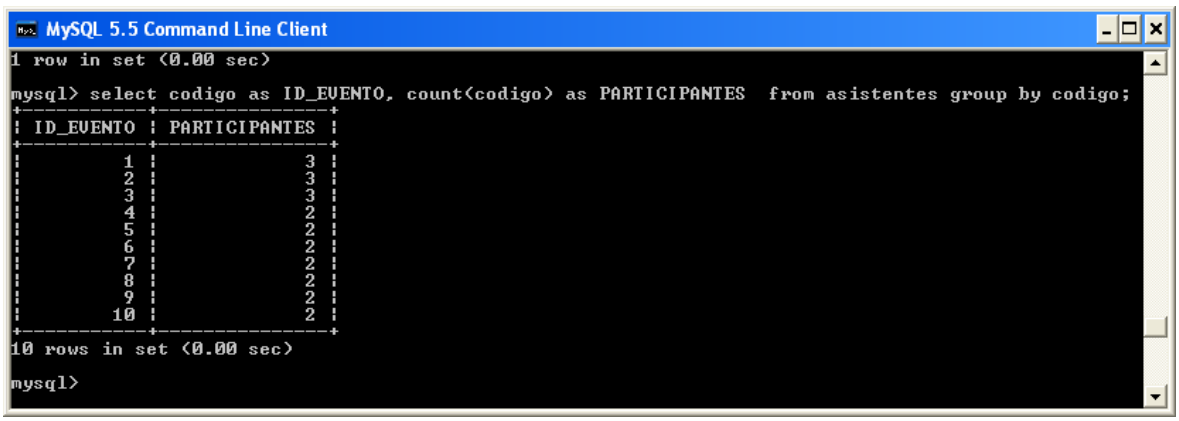
Se puede indicar la cantidad de resultados que se quieren mostrar añadiendo la cláusula LIMIT

2.5.5 Funciones SQL

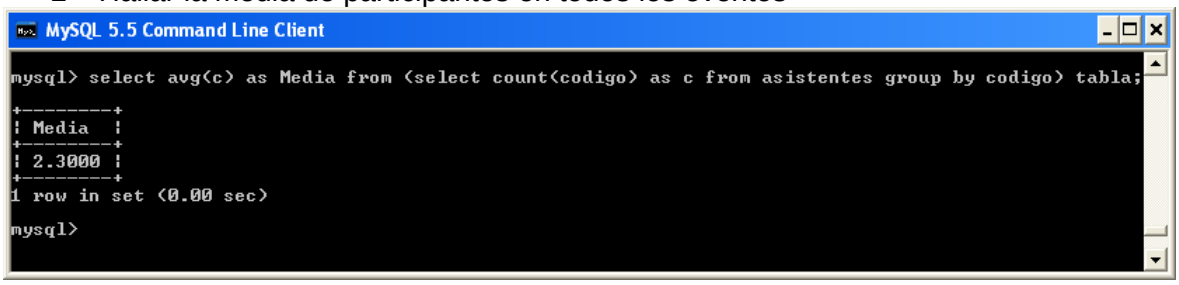
- AVG() devuelve la media de un atributo numérico
- COUNT() devuelve el número de columnas que coinciden con la restricción indicada
- GROUP BY() agrupa los resultados

Combinando estas funciones y SELECT anidadas podemos conseguir:

1- Contar el número de asistentes a cada evento



2- Hallar la media de participantes en todos los eventos



2.6 Estudio de herramientas de interpretación del lenguaje natural

En este capítulo se va a proceder a estudiar herramientas que interpretan el lenguaje natural y que si es posible lo usen para realizar consultas SQL

2.6.1 SQL-HAL: Natural Laguage to SQL Translator

Aplicación que traduce consultas en lenguaje natural a consultas SQL, las ejecuta y devuelve los resultados obtenidos.

Desarrollada en Perl, realiza análisis descendente para interpretar las consultas.

Puede interpretar consultas simples sobre una sola tabla y sobre dos tablas *"joined SQL".*

Solo implementa consultas SELECT. Una de las razones es porque las consultas select no modifican la base de datos, mientras que DELETE, INSERT, CREATE, etc. Sí que la modifican y es más difícil su control para asegurar la integridad de la base de datos.

Para poder realizar consultas con lenguaje natural SQ-HAL necesita saber la estructura de la base de datos, los nombres de las tablas, nombres de las columnas y los tipos de las columnas.

Usa *"Parse::RecDescent"* para implementar el analizador del lenguaje natural. Es un analizador de arriba a abajo

Parse::REcDescent

Realiza las funciones de analizador e interprete. La macro que usa para definir las reglas es el siguiente: Macro_name : macro_body

Ej:

```
#!/usr/bin/perl
  use Parse::RecDescent;
  # Create and compile the source file
  $parser = Parse::RecDescent->new(q(
    startrule : day  month /\d+/
    day : "Sat" | "Sun" | "Mon" | "Tue" | "Wed" | "Thu" | "Fri"
    month : "Jan" | "Feb" | "Mar" | "Apr" | "May" | "Jun" |
          "Jul" | "Aug" | "Sep" | "Oct" | "Nov" | "Dec"
  ));
  # Test it on sample data
  print "Valid date\n" if $parser->startrule("Thu Mar 31");
  print "Invalid date\n" unless $parser->startrule("Jun 31 2000");
```

Limitaciones y líneas de futuro

Como limitaciones se puede destacar que no detecta sinónimos, con lo que hay que referirse a los nombres exactos de las tablas y columnas. Solo es capaz de interpretar el inglés. Y necesita tener un conocimiento bastante amplio de la estructura de la BBDD.

Como líneas futuras quieren incorporar nuevas gramáticas e introducir más tipos de consultas, no solo SELECT

2.7 Definición del subconjunto del lenguaje natural

Puesto que abarcar el conjunto completo del lenguaje natural sería complicado. Se va a definir un subconjunto del lenguaje natural muy restrictivo, para posteriormente ir aumentándolo hasta donde sea posible.

2.7.1 Consulta SELECT

Las consultas podrían empezar por muchas palabras, pero básicamente comenzaremos con obligando a comentar las consultas por las palabras *"MOSTRAR"* o *"SELECCIONAR"*

2.7.2 Selección de campos y tablas

Para indicar los campos y las tablas que se quieren mostrar será necesario indicar sus nombres tal y como están definidos en la base de datos:

- Persona
    - DNI
    - Nombre
    - Apellido1
    - Apellido2
- Evento
    - Código
    - Fecha
    - Hora_ini
    - Hora_fin
    - Descripción
- Asistentes
    - Código
    - DNI
    - Confirmado

Posteriormente se podrá ampliar el reconocimiento de los campos como

- Apellido1 → Primer apellido
- Apellido2 → Segundo apellido
- Apellido1 y Apellido2 → Apellidos
- Hora_ini → Hora inicio, inicio, comienzo
- Hora_fin → Hora final, final

2.7.3 Clausula WHERE

Su sustituta inmediata es "QUE" y este será su identificador. Esta cláusula dará paso a las restricciones de la cláusula WHERE

Operadores de la cláusula WHERE

La identificación de los operadores será la siguiente

Operador	Descripción
=	Igual
<> (o !=)	Distinto
>	Mayor que
<	Menor que
>=	Mayor o igual que
<=	Menor o igual que
BETWEEN	entre
LIKE	Como
IN	Incluido en

Los operadores AND y OR serán identificados por *"Y"* y *"O"* respectivamente

2.8 Costes y beneficios del proyecto

2.8.1 Estimaciones de costes

Las herramientas proporcionadas son de software libre, como por ejemplo Jlex y CUP, NetBeans y MySQL o han sido proporcionadas por la universidad a lo largo de la carrera, como Microsoft office, Microsoft proyect

Por lo que solo estimaremos el coste del desarrollo

	Horas	**Euros**
Captación de requisitos	30	360
Análisis	60	720
Implementación	150	1800
Pruebas	30	360
Total	**270**	**3240**

2.8.2 Beneficios

Aunque la herramienta sea una prueba de concepto o introducción al procesamiento del lenguaje natural. Es funcional y tiene algunos usos. Como por ejemplo educacional. Para ayudar a comprender el lenguaje SQL y ayudar a demostrar la relación que tiene con el lenguaje natural.

Aun así la herramienta no es suficiente todavía para considerarlo un producto final y lanzarlo al mercado.

3 DISEÑO TÉCNICO E IMPLEMENTACIÓN DEL SISTEMA

3.1 Modelo de desarrollo

La primera parte del proyecto constara de recogida de información, estudio de herramientas, documentación. Para adquirir todos los conocimientos necesarios para el desarrollo de la herramienta final.

En esta parte también se definirán puntos importantes del desarrollo de la herramienta. Tales como el conjunto permitido de lenguaje natural y el alcance de las sentencias SQL.

La segunda parte se desarrollara una interfaz sencilla en java donde se proporcione un campo de entrada para realizar las consultas, un campo de salida donde se mostrara las respuestas y un campo donde se mostraran los errores y posibles soluciones.

Se pretende abstraer al máximo posible la herramienta de la interfaz gráfica, ya que la idea es que la herramienta sea fácil de portar a cualquier escenario.

La herramienta está dividida en:

- **Programa base** que recibe una sentencia en lenguaje natural y lo traduce a sentencia SQL. Este consta de las siguientes partes:
    - **Analizador léxico:** Recibirá el texto introducido en la interfaz gráfica, reconocerá los *tokens* y los enviará al analizador sintáctico
    - **Analizador sintáctico:** Recibirá los *tokens* del analizador léxico y verificara la corrección de la estructura conforme a la gramática definida
    - **Analizador semántico:** Se encargara de verificar el correcto significado de la estructura
    - **Generador de errores y alternativas:** A medida que se va tratando el texto de entrada por los analizadores, se irán generando posibles errores y alternativas a este.
- **Interfaz gráfica** que:
    - hace uso del programa base para transformar la sentencia en lenguaje natural a sentencia SQL
    - Realiza la consulta SQL a la base de datos
    - Muestra los resultados en su entorno.

3.2 Entorno de desarrollo

El programa base será desarrollado con JLEX y CUP. Estas herramientas nos permiten hacer un producto a medida, pero también el coste de desarrollo es más elevado. Esto es debido a que son unas herramientas de bajo nivel de programación. Con lo que hay que desarrollar todas las funcionalidades desde cero. Por el contrario, ofrece muy pocas restricciones a la hora de definir nuestras funcionalidades. Ya que no se hace uso de módulos ya creados que tengan sus propias funcionalidades que tendríamos que intentar adaptar a las nuestras.

3.2.1 JLEX y CUP

La programación de JLex y CUP se ha desarrollado en un editor de texto plano. Ya que actualmente no hay ninguna herramienta de desarrollo java que implemente correctamente el uso de ellas.

La estructura de un fichero JLex es la siguiente

- Sección de código de usuario. Se copia directamente en el explorador que se genera al compilar JLex y CUP. En esta sección se incluyen sentencias java de importación y la definición de clases y tipos de datos necesarias para la aplicación.
- La Sección de directivas. Se utiliza para definir características del explorador generado y donde se declaran los macros y estados usados en la definición de las reglas
- Sección de reglas de expresiones regulares. Contiene las reglas que identifican como tokens las agrupaciones de caracteres.

La estructura de un fichero CUP es la siguiente

- Especificaciones de importación y empaquetado
- Código de usuario. Declaraciones que permiten incluir código en el analizador generado
- Lista de símbolos. Se declaran los terminales y no terminales utilizados en la gramática. También se puede incluir el tipo de objeto que va ligado a ellos
- Declaraciones de precedencia
- Gramática. Definición de las reglas de la gramática.

Por ultimo indicar que la herramienta se ha definido como una clase Java estática sin métodos de creación. Tan solo un método en el que se pasa como entrada el texto a

procesar y devuelve la sentencia SQL y un segundo método que devuelve los conjuntos de caracteres no utilizados en el último procesamiento de texto

3.2.2 Base de datos MySQL

Sistema de gestión de bases de datos relacional, multihilo y multiusuari.

Se ha utilizado por ser un entorno ya conocido, de software libre, fácil instalación y cubrir las necesidades del proyecto.

La base de datos es sencilla y se compone de las siguientes tablas y relacionse

- Tabla Persona, con los campos DNI, Nombre, Apellido1 y Apellido2. Siendo El DNI la clave primaria
- Tabla Evento con los campos Código, Fecha, Hora de inicio, Hora de fin y descripción. Siendo el código la clave primaria
- Tabla Asistentes con los campos Código, DNI, confirmado. Siendo Código y DNI la clave primaria y a su vez claves foráneas de las anteriores tablas.

3.2.3 Interfaz gráfica de usuario

El desarrollo de la interfaz gráfica de usuario se ha desarrollado por medio de la herramienta NetBeans. Actualmente hay dos herramientas muy potentes para el desarrollo de java (Eclipse y NetBeans) con pocas diferencias entre ellas. Por lo que no hubo un motivo reseñable para la elección.

La herramienta facilita la importación de librerías, la programación en lenguaje Java y el diseño del entorno gráfico. Ya que solo hay que arrastras los paquetes contenedores de los gráficos al entorno de diseño y darles las dimensiones y apariencia necesaria.

Para poder hacer uso de la herramienta de conversión de sentencias de lenguaje natural en consultas SQL. Se definió esta de tal manera que fuera fácil su importación como una librería para poder invocar sus métodos.

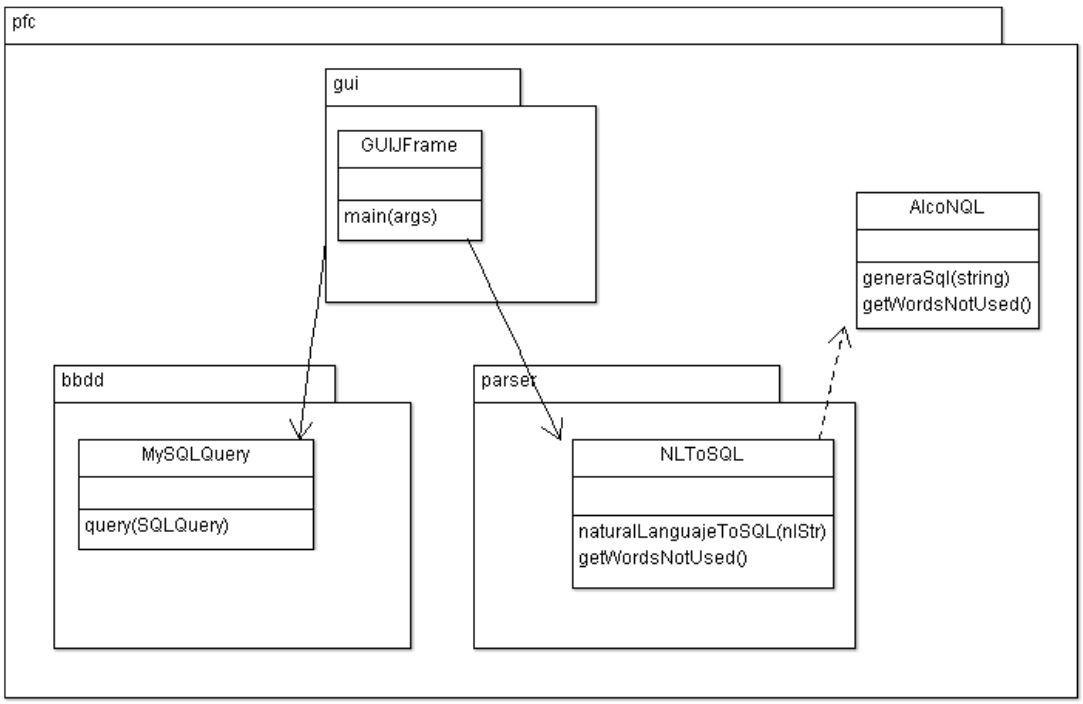
Para la realización de llamadas a la base de datos se usó la herramienta de conexión a base de datos JDBC por la que es trivial establecer la conexión y realizar llamadas.

3.3 Diagramas de diseño

3.3.1 Diagrama de caso de uso

Nombre	**Transformar lenguaje natural en sentencia SQL**
Resumen de la funcionalidad	Permite al usuario transformar una sentencia en lenguaje natural en lenguaje SQL y visualizar los resultados de la base de datos
Flujo del acontecimiento principal	1. El usuario introduce la sentencia en lenguaje natural y ejecuta la acción transformar 2. El sistema muestra la sentencia SQL correspondiente y el resultado de la consulta a la base de datos. En la pestaña “Palabras no usadas” se muestran las palabra que no han sido utilizadas en la transformación
Flujo de acontecimientos alternativos	2ª. Si la consulta no puede ser transformada se muestra el siguiente error “Error: estructura SELECT no encontrada” y no se muestran datos de la base de datos 2ᵇ. Si la consulta SQL produce un error en la base de datos se muestra el error

3.3.2 Diagrama de clases



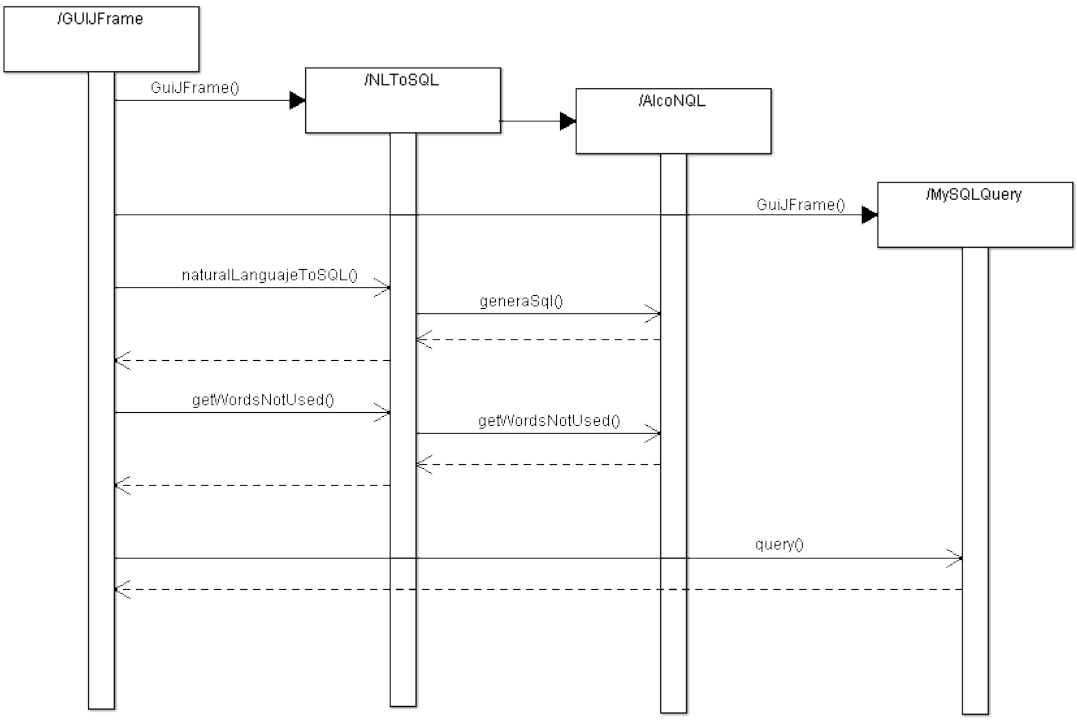
GUIJFrame: Clase encargada de mostrar el la interfaz gráfica de la aplicación y controlar el uso del resto de clases. Tiene definidos los objetos de la interfaz gráfica como botones campos y tablas y los métodos para invocar a los servicios del resto de clases

NLToSQL: Clase encargada de gestionar las llamadas a AlcoNQL. Tiene definidos los métodos de invocación a los servicios de conversión de sentencias en lenguaje natural a consultas SQL y recuperación de conjunto de caracteres no utilizados de la clase AlcoNQL

AlcoNQL: Herramienta de transformación de sentencias en lenguaje natural a SQL desarrollada en JLex y Cup. Clase estática sin método de creación. Tan solo cuenta con el método de transformación de sentencias en lenguaje natural a consultas SQL y recuperación de los conjuntos de caracteres no utilizados en la última consulta

MySQLQuery: Clase encargada de realizar las consultas SQL a la base de datos utilizando la api de java JDBC

3.3.3 Diagrama de secuencia



El proceso normal que sigue la herramienta es el siguiente

1. Al abrir la aplicación se crean los objetos como se indica en en el diagrama
2. Se introduce el texto en la interfaz gráfica de usuario y se ejecuta el proceso
3. Al iniciarse el proceso se realiza la llamada al objeto de la clase NLToSQL. Clase controladora de la herramienta de transformación de sentencias en lenguaje natural al consultas SQL
4. Esta a su vez realiza la petición de transformación de la sentencia en lenguaje natural a consulta SQL al objeto de la clase AlcoNQL. Herramienta de transformación de sentencias en lenguaje natural a consultas SQL
5. El objeto AlcoNQL devuelve la respuesta del procesamiento del lenguaje natural al objeto NLToSQL y este a su vez lo devuelve al objeto principal
6. Se repite el mismo proceso invocando el método que devuelve los conjuntos de caracteres no utilizados en el proceso anterior
7. Una vez obtenida la consulta SQL se realiza la petición de consulta a la base de datos realizando la llamada al objeto MySQLQuery que devuelve la respuesta realizada por la base de datos a la clase principal

3.4 Implementación de la herramienta

3.4.1 JLex

En el analizador léxico se han definido los siguientes patrones de caracteres

Identificadores de SELECT

Como comienzo de la frase y a su vez identificadores de la palabra clave SELECT se han definido las palabras:

- Mostrar
- Muestra
- Selecciona

Identificadores de atributos de las tablas

Para cada uno de los atributos de las tres tablas se ha definido un patrón de caracteres que lo identifica:

- DNI
- Nombre
- Apellido1
- Apellido2
- Código
- Confirmado
- Fecha
- Hora inicio
- Hora fin
- Descripción

Algunos tienen más de una variante para acercarlos más al lenguaje natural:

- Apellido1 → Apellido o Primer apellido
- Apellido2 → Segundo apellido
- Confirmado → Confirmados

También se han definido los patrones todo, toda, todos, todas para seleccionar todos los atributos

Identificadores de tablas

Se ha definido un identificador para cada tabla y su plural

- Persona o Personas
- Evento o Eventos
- Asistente o asistentes

Identificador de cláusula WHERE

Se identifica con los siguientes patrones:

- Que
- De
- Con

Cláusulas de comparación

- “>” → mayor que
- “<” → menor que
- “>=” → mayor o igual que
- “<=” → menor o igual que
- “=” → igual o igual a
- “<>” → distinto o distinto de
- “between” → entre
- “in” → en o incluidos en

Cláusulas de unión

- Y → y
- O → o

Identificadores

Hay cuatro tipos de identificadores

- Números → cualquier cadena de números
- Identificadores → cualquier combinación de caracteres del alfabeto junto al guion bajo
- Hora → formato HH:MM siendo H: hora, M: minuto
- Fecha → formato YYYY-MM-DD siendo Y: año, M:mes, D:dia

Estados

Hay definidos 6 estados para poder acercarse más al lenguaje natural

- Estado inicial (YYINITIAL): se controla el inicio de la consulta y la cláusula WHERE
- Estado COLUMNAS: se controla los atributos a mostrar y la tabla seleccionada
- Estado WHERE: Se controla la conjunción dentro de la cláusula, atributos por los que se va a definir la cláusula y comparadores.
- Estado WHEREVALUE: en él se captan los valores de los atributos en la cláusula WHERE, también se permite que pueda haber peticiones recursivas.
- Estado WHEREBETWEENVALUE: Se captan los valores de la cláusula BETWEEN
- Estado WHEREINVALUE: se captan los valores de la cláusula IN, también se permite que pueda haber peticiones recursivas

Restricciones

Debido al intento de captar correctamente los valores sin el uso de delimitadores. Se ha tenido que sustituir el espacio por el guion bajo a la hora de formar identificadores. Quedando de la siguiente manera:

Ej: cuando un identificador sea una palabra compuesta como CENA DE NAVIDAD. Esta se indicará como CENA_DE_NAVIDAD

Esta restricción es debida a haberle dado a los espacios el uso de delimitadores de los identificadores para acercarnos más a la interpretación del lenguaje y evitar así delimitadores estilo comillas para indicar a los identificadores.

3.4.2 CUP

Se han definido las siguientes consultas por medio de reglas.

Selección de campos de una tabla

Muestra todos los campos y registros de una tabla

- *TK_SELECT colums:s1 table:s2*

Siendo los posibles campos cada uno de los campos disponibles en las tablas, o todos ellos. Si no se indica ningún campo se considera que hay que mostrar todos

Clausula WHERE asociada a una SELECT

Su sintaxis normal es la de IDENTIFICADOR COMPARADOR VALOR

- *where ::= atrib:s1 TK_IN select:s2*

O la sustitución de un atributo por una clausula

- *where ::= atrib:s1 comp:s2 select:s3*

Se controla la unión de cláusulas con los tokens Y y O

- *where2 ::= where2:s1 TK_Y where2:s2 | where2:s1 TK_O where2:s2*

Pueden englobarse las clausulas entre paréntesis

- *where2 ::= TK_I_PAR where2:s1 TK_D_PAR*

Pueden incluirse clausulas recursivas

- *where2 ::= atrib:s1 comp:s2 select:s3*

Clausula WHERE BETWEEN: indicando los dos valores separados con el token Y

- *where ::= atrib:s1 TK_BETWEEN TK_VALOR:s2 TK_Y TK_VALOR:s3*

Clausula WHERE IN: se pueden incluir tantos valores como se estime oportuno u otra consulta SELECT recursiva

- *whereIn ::= whereIn:s1 TK_VALOR:s2 | TK_VALOR:s2*
- *where ::= atrib:s1 TK_IN select:s2*

3.4.3 Control de errores

- Se recogen todas las palabras no usadas y se obtienen usando el método getWordsNotUsed
- Si una consulta no se acoge sintácticamente a las reglas definidas en vez del resultado se indica el error

Se han definido una serie de reglas que representan las estructuras de consultas SQL permitidas en Cup. Si la consulta recibida no se acoge a ninguna de las reglas definidas se acogerá a la regla de error.

Esta regla no hace ningún tipo de tratamiento de la estructura definida. Por lo que solo devolverá un mensaje de error.

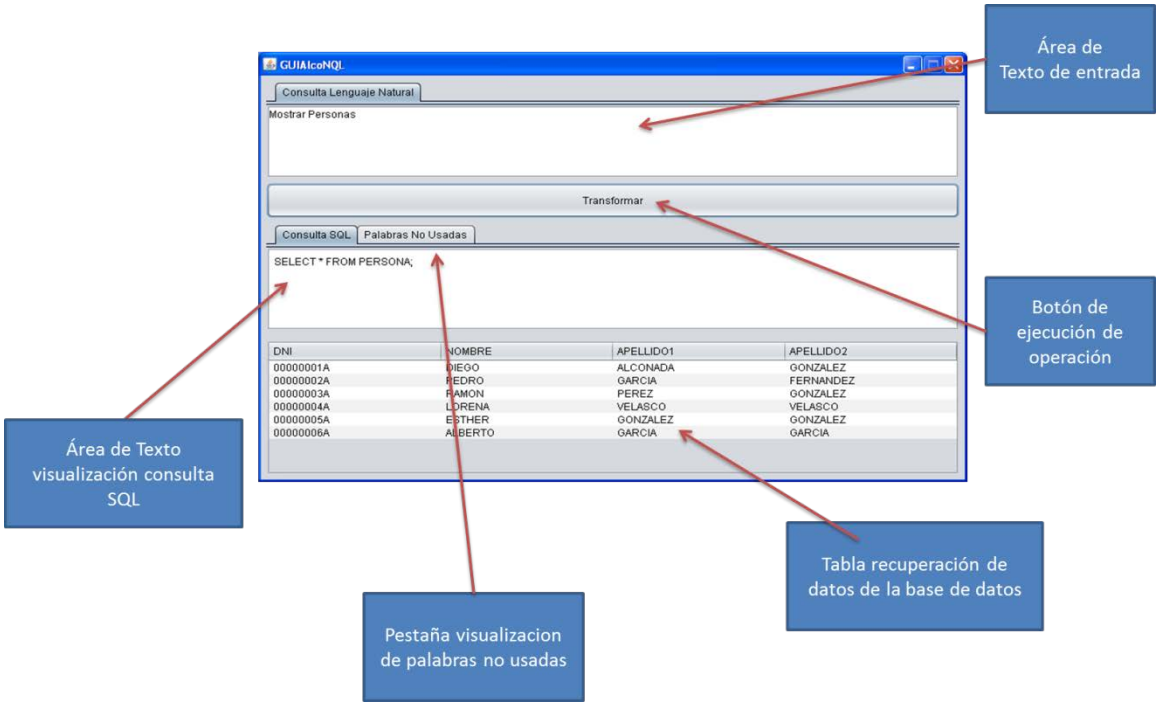
Una posibilidad para el tratamiento de errores sería definir reglas para cada tipo de error y tratarlas como las reglas definidas para las estructuras correctas. Y si alguna secuencia se acoge a esta regla dar como respuesta las palabras que faltan o sugerir una pregutna

Por ejemplo, la regla principal para la identificación está conformada de la siguiente manera, indicando entre <> los que son opcionales

Token_SELECT <Tokens_Columnas> Token_Tabla

Teniendo esta regla se podrían definir reglas en las que faltara el Token_SELECT o El Token_Tabla, con lo que estarían incompletas, sabríamos el token que falta y podríamos sugerirlos.

3.4.4 Definición de la interfaz gráfica de usuario



3.4.5 Dependencias y relaciones entre los módulos de la herrameinta

La herramienta se ha hecho lo más genérica posible, para que sea lo más transparente a los cabios que ha sido posible.

Si se quiere añadir nuevas bases de datos, tablas o campos en las tablas habrá que añadir los tokens correspondientes tanto al JLex como al CUP. Tanto el JLex como el CUP están correctamente estructurados para que los cambios sean fácilmente aplicables. Están claramente definidas las estructuras de tablas y columnas con lo que solo habría que añadir las nuevas modificaciones en ellas.

Si se quieren añadir nuevo módulos para interactuar con la herramienta como reconocedores de voz. Tan solo habría que añadir el reconocedor de voz a la herramienta gráfica y dar acceso a el modulo. Y sustituir la entrada de texto por la entrada de voz. Este módulo de voz devolverá un texto que será el que hay que dar como entrada a la herramienta de transformación de lenguaje natural a SQL

4 TEST Y PRUEBAS

4.1 Fase de pruebas

Este proyecto es una introducción y prueba de concepto de la necesidad de realizar consultas a base de datos de por medio del lenguaje natural. No se puede considerar un producto final. Con lo que, aunque el producto sea funcional tiene tareas pendientes como la mencionada punto 3.4.3 control de errores. Por la que se pueden controlar errores de falta de palabras, sugiriendo consultas correctas.

Se han ido realizando pruebas unitarias a la par que la implementación del producto. Una vez completado un módulo y sus pruebas unitarias. Se ha hecho pruebas de su integración en el producto.

4.1.1 Pruebas de la base de datos

Requerimiento	Resultado esperado	Resultado obtenido
Acceso a la base de datos a través de la herramienta MySQL	Acceder correctamente desde la herramienta mysql.exe a la base de datos	Correcto
Conexión a la base de datos desde java	Conexión a la base de datos desde el driver ojdbc	Correcto
Ejecucion de las consultas SQL desde java	Ejecucion de consultas SQL desde java por medio del driver ojdbc	Correcto

4.1.2 Pruebas herramienta grafica

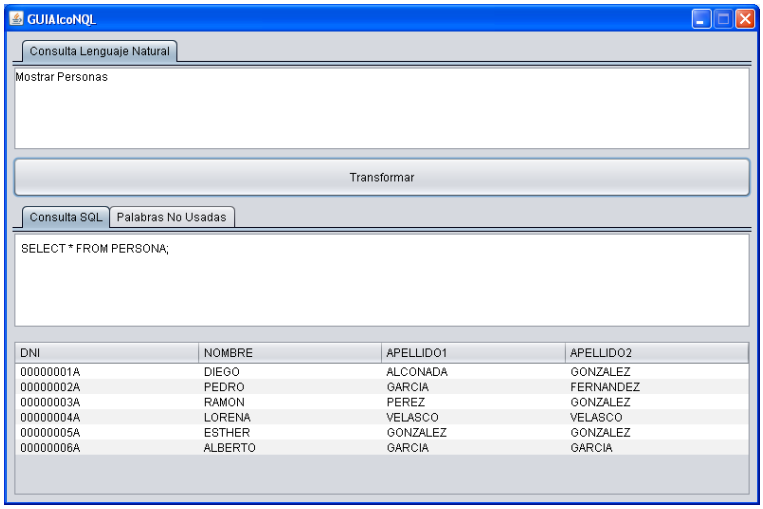
Requerimiento	Resultado esperado	Resultado obtenido
Acceso a herramienta de transformación de sentencias	Uso de la librería de transformación de sentencias dando como entrada una sentencia en lenguaje natural y obteniendo como resultado la sentencia en SQL o el error producido y las palabras no usadas	Correcto
Envió de consultas SQL a la base de datos	Envió de las sentencias SQL producidas por la herramienta de transformación de sentencias a la base de datos y obtención de la respuesta	Correcto
Visualización de la herramienta grafica	Visualización de los campos de entrada y las respuestas correctamente	Correcto

4.2 Ejemplos de ejecución

A continuación se muestras diversas consultas a realizar

Consulta básica

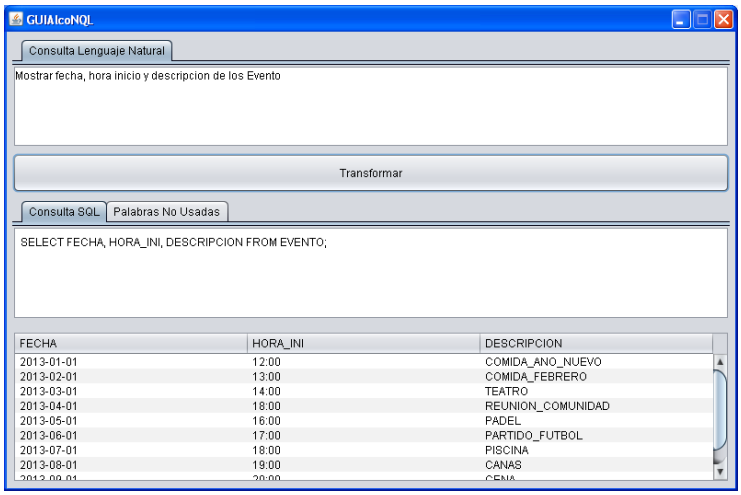
- MOSTRAR PERSONAS
- MUESTRA EVENTOS
- SELECIONA ASISTENTES



Consulta de campos de una tabla

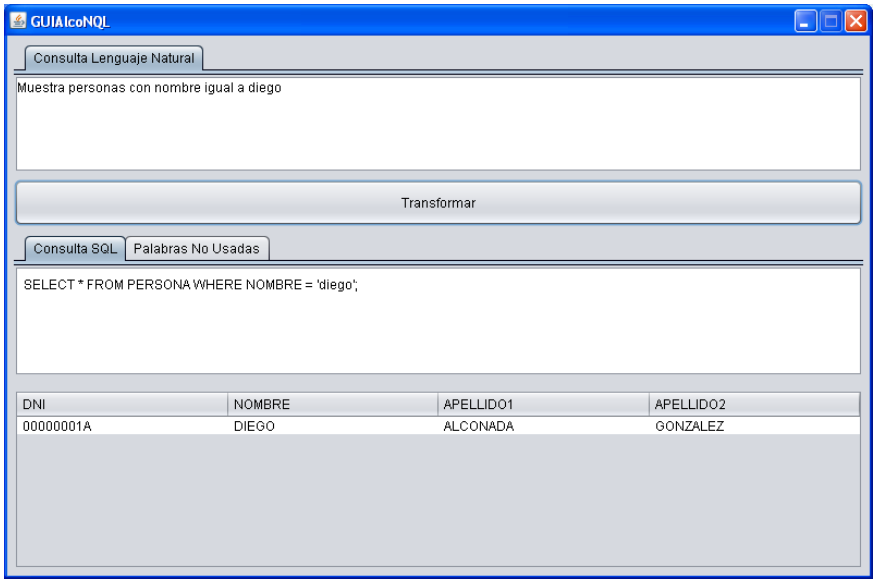
Consulta

- MOSTRAR FECHA, HORA INICIO Y DESCRIPCION DE EVENTOS
- Cualquier combinación de los atributos de cada tabla, pudiendo incluir conjunciones entre ellas

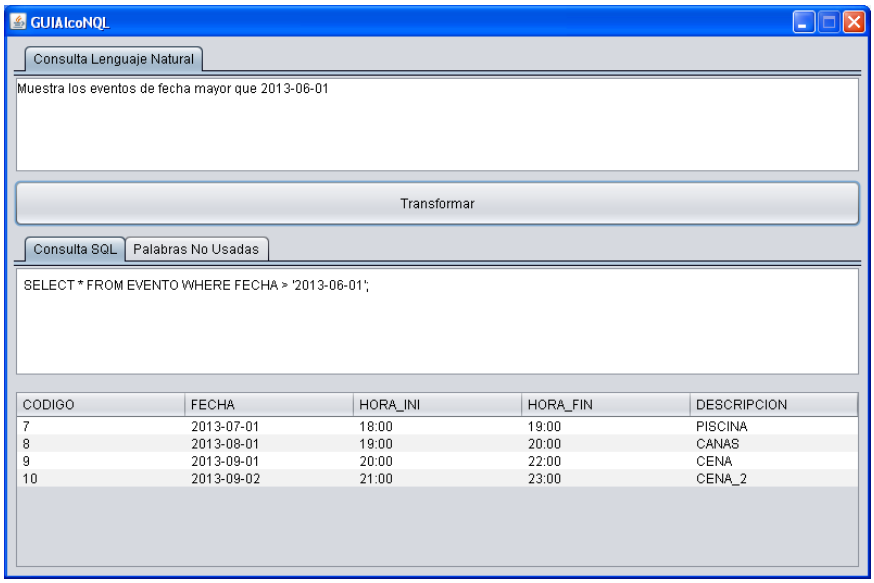


Consulta clausula WHERE

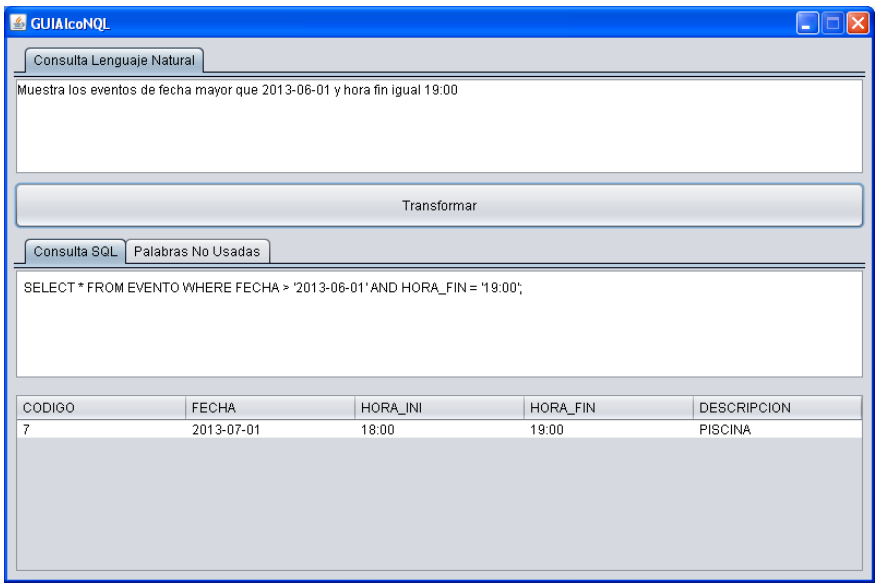
- MUESTRA PERSONAS CON NOMBRE IGUAL A DIEGO



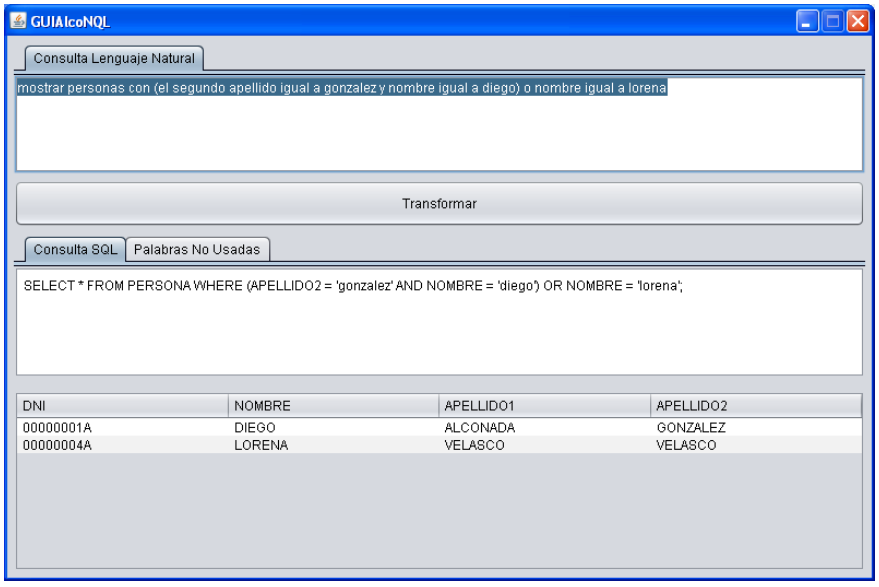
- MUESTRA LOS EVENTOS DE FECHA MAYOR QUE 2013-06-01



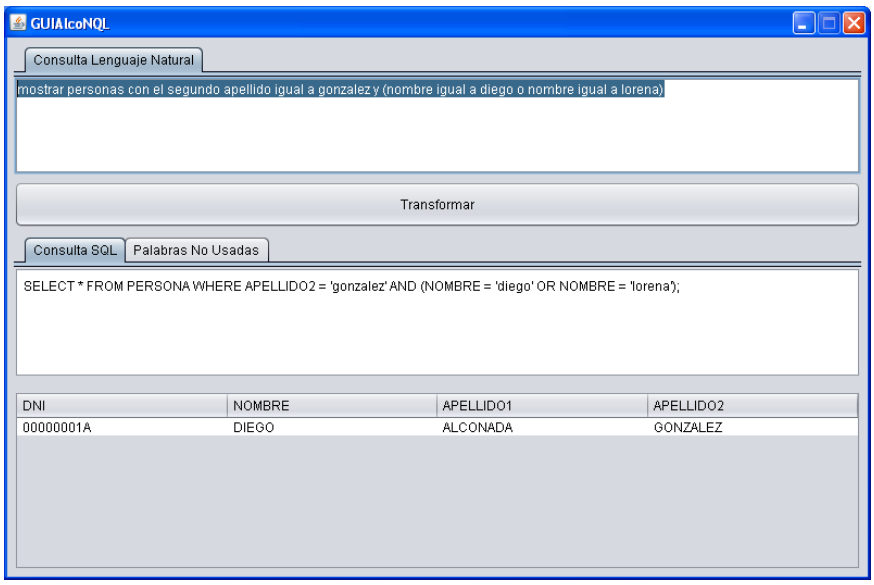
- MUESTRA LOS EVENTOS DE FECHA MAYOR QUE 2013-06-01 Y HORA FIN IGUAL 19:00



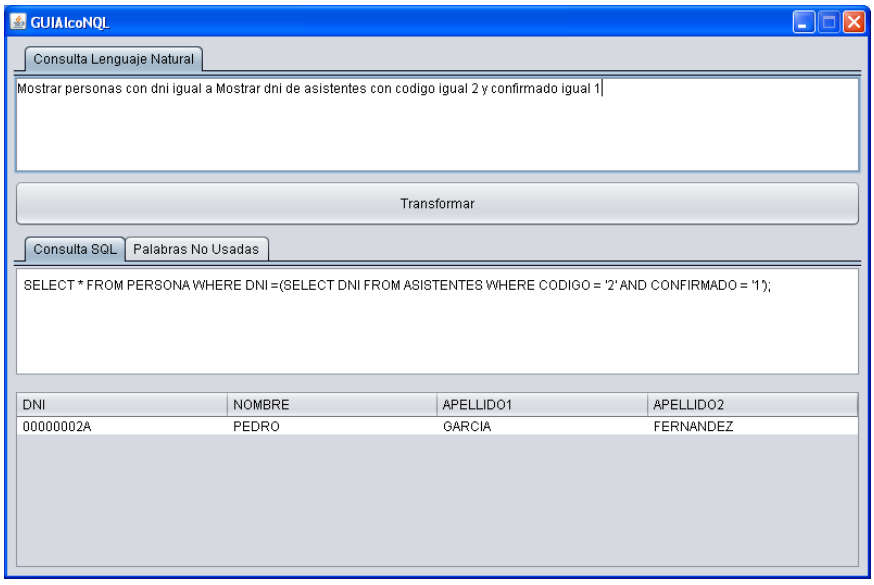
- Se pueden añadir tantas condiciones como se quieran unidas por medio de “Y” u “O” y anidarlas entre paréntesis para indicar su precedencia. De esta forma tendremos estas variantes
- Mostrar personas con (el segundo apellido igual a González y nombre igual a Diego) o nombre igual a Lorena



- Mostrar personas con el segundo apellido igual a Gonzalez y (nombre igual a Diego o nombre igual a Lorena)

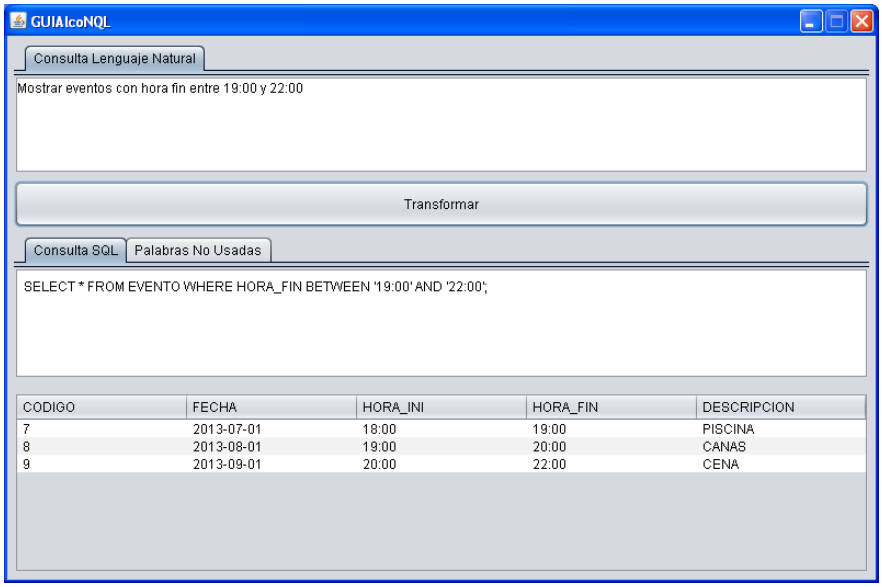


- También se pueden hacer comparaciones con otras consultas
- Mostrar personas con dni igual a Mostrar dni de asistentes con codigo igual 2 y confirmado igual 1



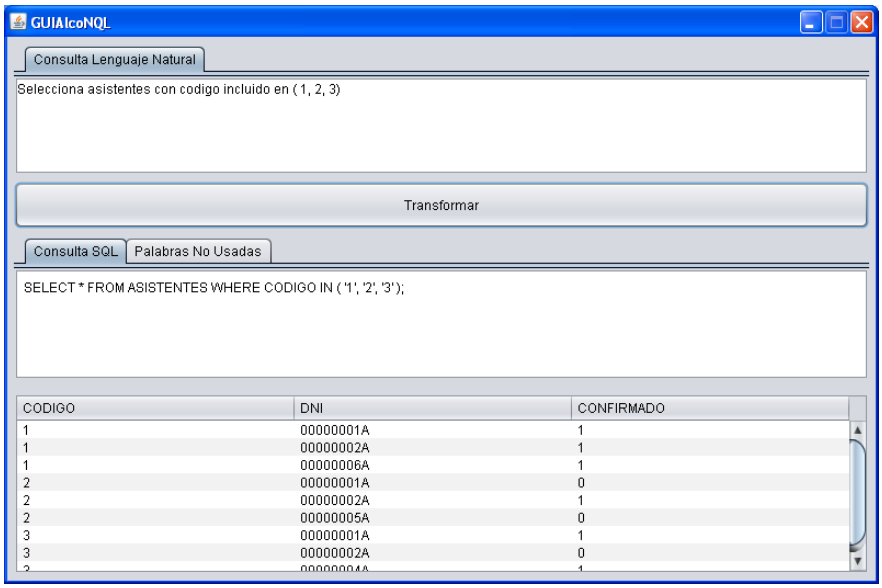
Consulta clausula WHERE BETWEEN

- Mostrar eventos con hora fin entre 19:00 y 22:00



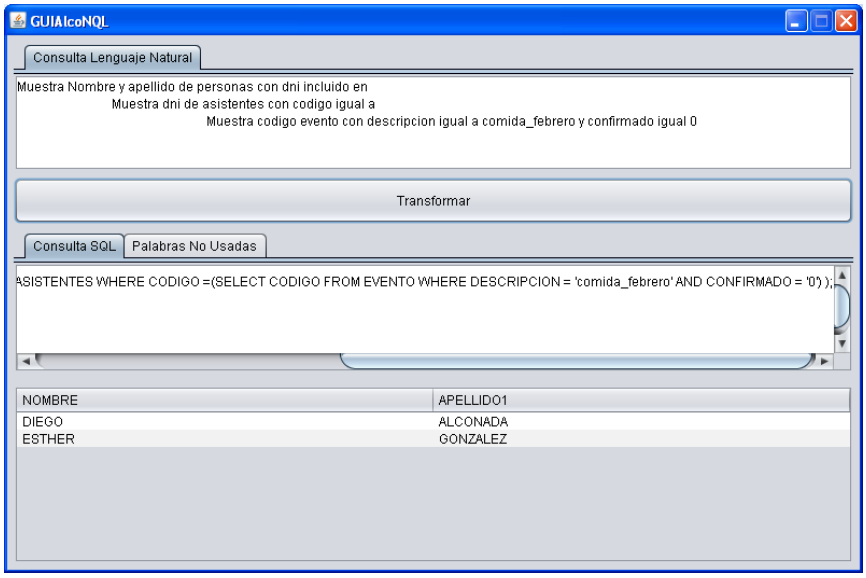
Consulta clausula WHERE IN

- Selecciona asistentes con código incluido en (1, 2, 3)



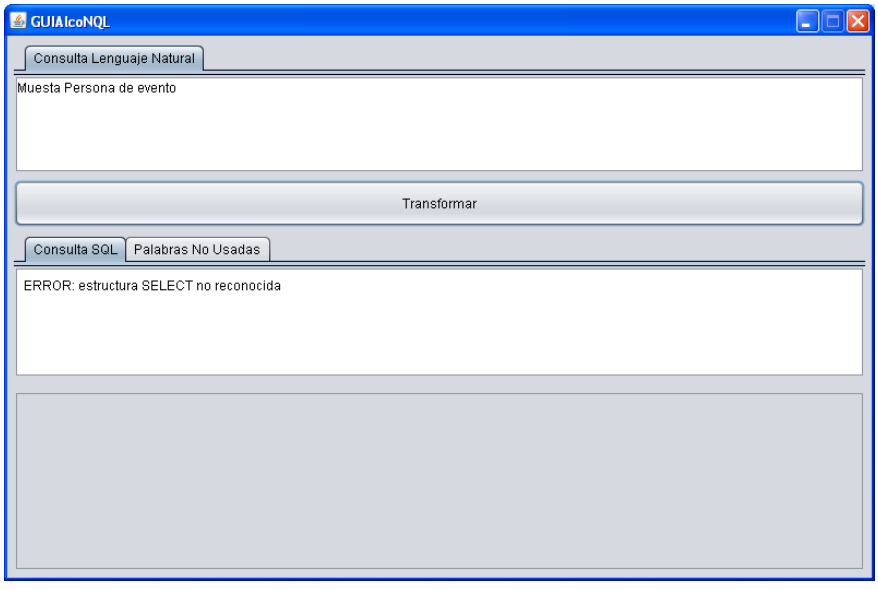
Consulta compleja

- Muestra Nombre y Apellido de personas con DNI incluido en Muestra DNI de asistentes con codigo igual a Muestra codigo evento con descripcion igual a comida_febrero y confirmado igual 0

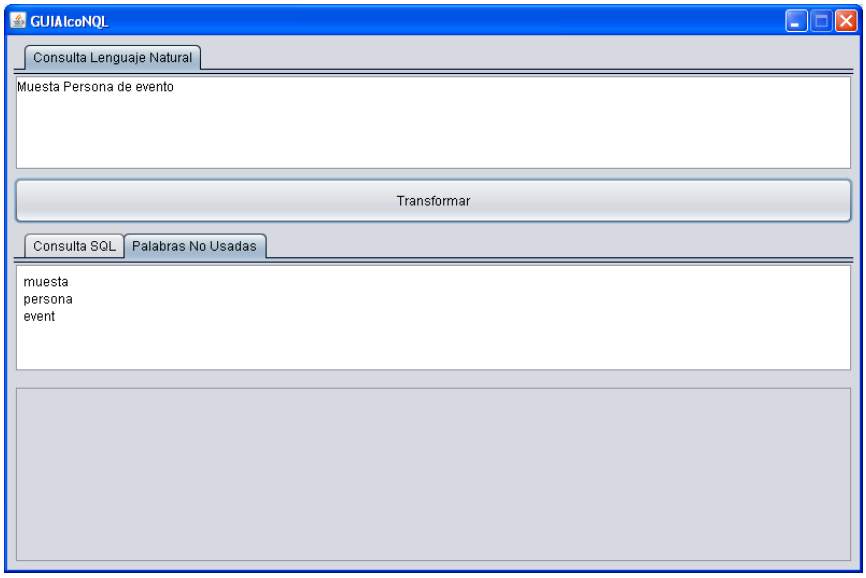


Muestra de errores

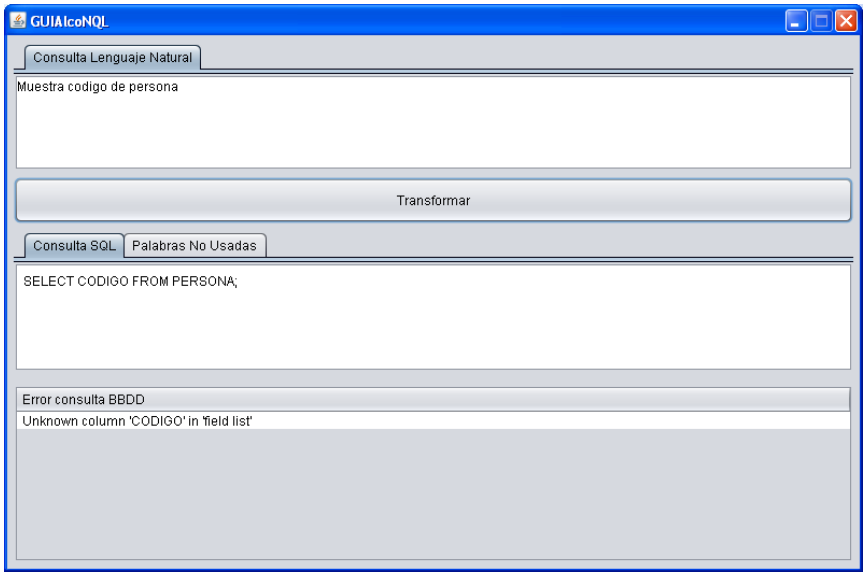
- Estructura no reconocida, Indica el error en la consulta SQL



Y en la pestaña de palabras no usadas se puede comprobar Que no hemos escrito correctamente la palabra MUESTRA y ya no reconoce ninguna de las siguientes palabras



- Error en la consulta a la base de datos, En la tabla se muestra el error recibido de la base de datos



5 LÍNEAS FUTURAS Y CONCLUSIONES

5.1 Líneas futuras

A continuación se listas posibles líneas futuras:

- Sustitución de la base de datos pos una que pueda integrarse completamente en la aplicación y no requiera de instalar un servidor, por ejemplo HSQLDB
- Implementación de mayores funcionalidades de la sentencia SELECT. Como por ejemplo la unión de tablas
- Implementación de resto de sentencias SQL (INSERT, UPDATE, DELETE, CREATE…)
- Mejorar la captación de valores. Pudiendo reconocer valores que contengan espacios.
- Ampliación de las funcionalidades de la herramienta introduciendo reconocedores de voz como Siri o Dragon
- Desarrollo de una herramienta grafica que represente la base de datos, pudiendo aplicar esta información a la herramienta dinamicamente

5.2 Conclusiones

La realización de este proyecto ha ayudado a fortalecer los conocimientos adquiridos a lo largo de la carrera y una gran experiencia final para acercarnos al mundo laboral.

Los objetivos principales han sido cumplidos. Según se ha ido realizando el proyecto se han ido aprendiendo nuevas metodologías o mejores soluciones para aplicar al proyecto, que por falta de experiencia no se detectaron en el análisis. Estas soluciones no se han podido aplicar por falta de tiempo.

Aunque no se puede considerar un producto final. Si se puede pensar en los posibles usos que se le podrían dar a la herramienta.

- El primero y más claro, serian educacional. Es una buena herramienta para dar a conocer las bases de datos y ayudar a entender cómo funciona la consulta de las mismas.
- Toda necesidad que pueda requerir o se pueda implementar una base de datos. Por ejemplo, La carta de un restaurante, Aplicación móvil para buscar comercios o empresas cerca de tu localización, etc.

6 BIBLIOGRAFÍA

6.1 Apuntes

- Apuntes de la asignatura Ingeniería del software orientada a objetos de la universidad oberta de Cataluña
- Apuntes de las asignaturas de compiladores de la universidad oberta de Cataluña
- Manual JLex y CUP de la asignatura compiladores II de la universidad oberta de Cataluña

6.2 Libros

- SPEECH AND LANGUEGE PROCESSING (Second Edition) de Daniel Jurafsky y James H. Martin

6.3 Documentación Web

- NetBeans - http://netbeans.org/
- JLex - http://jflex.de/
- CUP - http://www2.cs.tum.edu/projects/cup/
- MySQL - http://www.mysql.com/
- JDBC - http://dev.mysql.com/usingmysql/java/
- SQ-HAL - http://www.csse.monash.edu.au/hons/projects/2000/Supun.Ruwanpura/
- FreeLing 3.0 - http://nlp.lsi.upc.edu/freeling/

7 ANEXOS

7.1 Instalación del entorno

Partimos de una instalación Windows XP actualizada

7.1.1 Instalación de la base de datos

Se ha elegido la base de datos MySQL por ser de libre distribución y cubrir las necesidades del proyecto.

Se descarga y ejecuta la versión indicada para el sistema operativo desde:

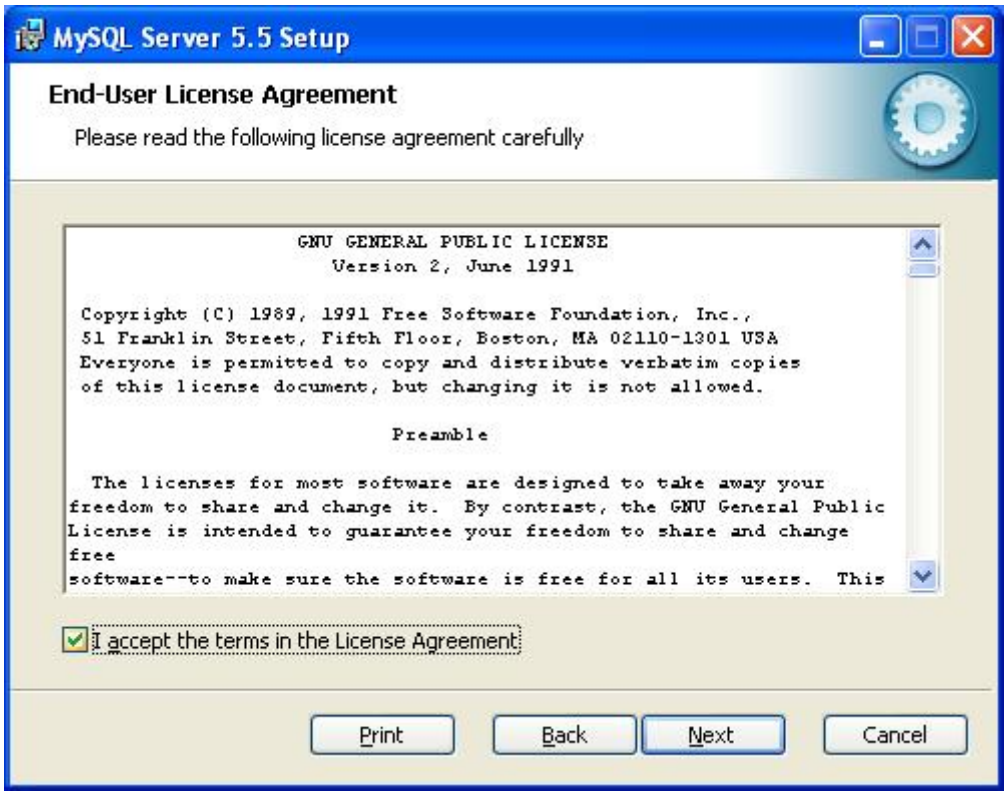
http://www.mysql.com/downloads/mysql/

En este caso se va a instalar la versión mysql-5.5.28-win32

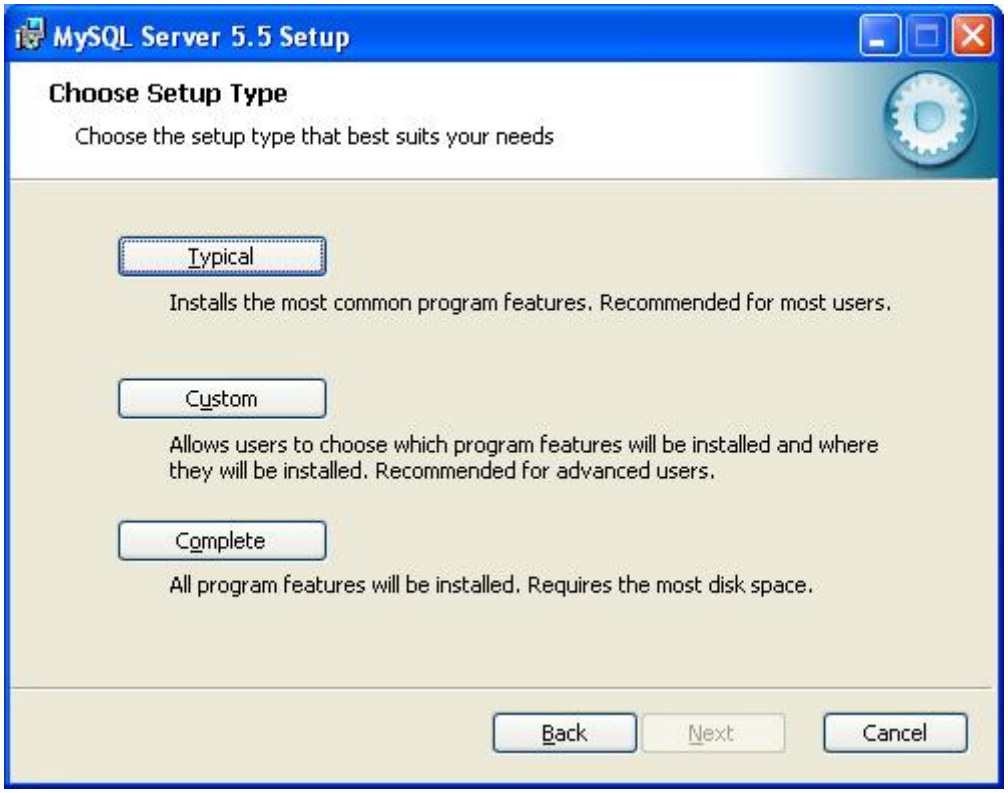
Los pasos de configuración son los siguientes

- Pantalla de inicio de la instalación

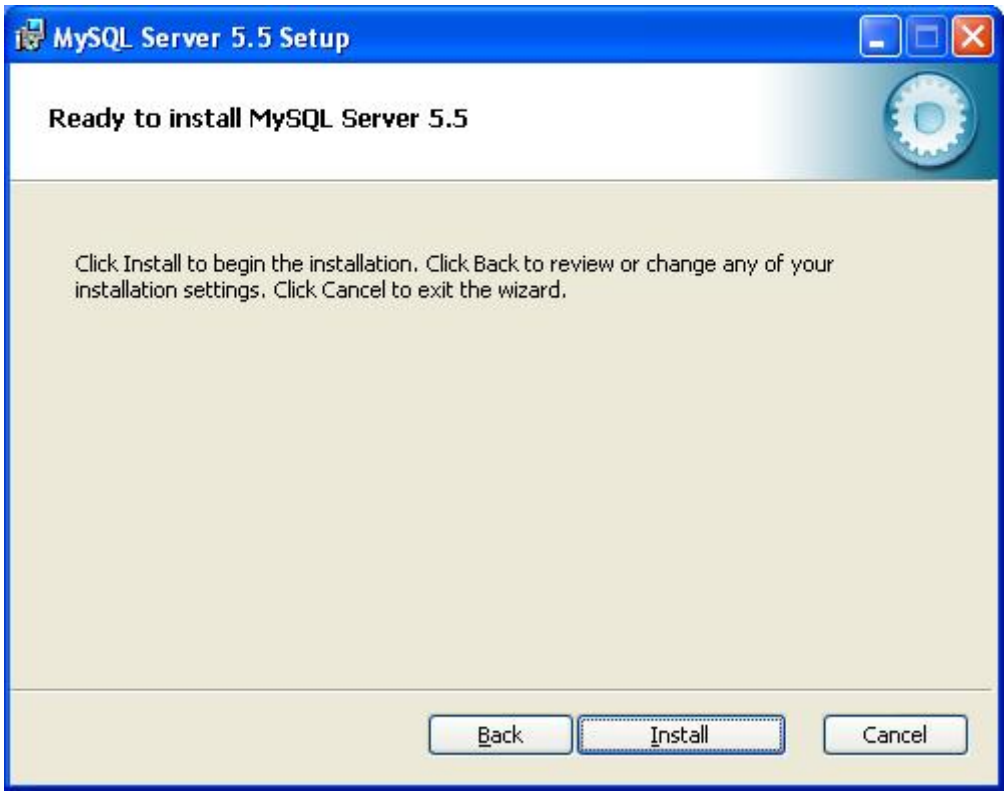
- Aceptación de licencia



- Selección de tipo de instalación. En este caso la completa

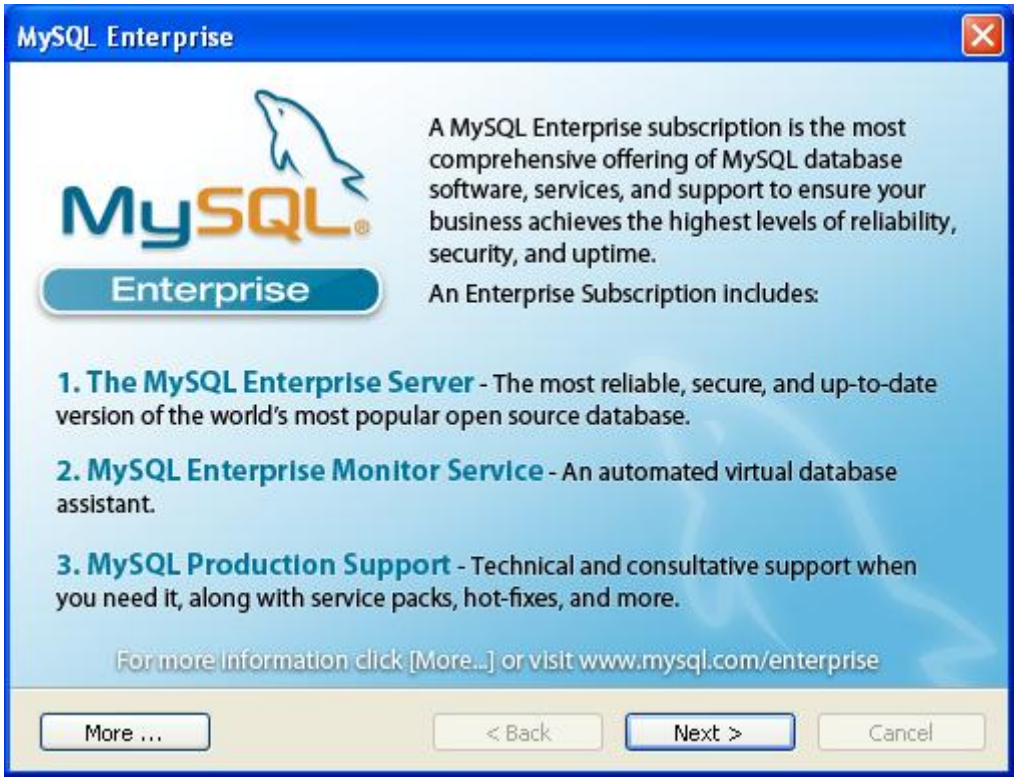


- Pantalla de confirmación de instalación



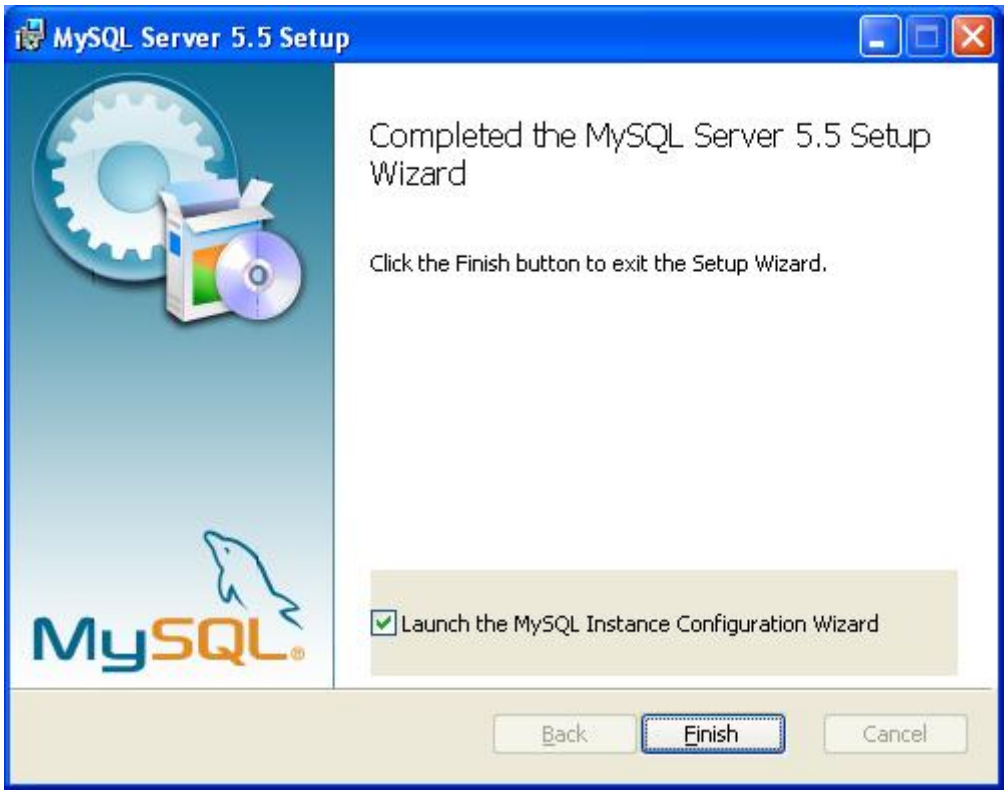
Una vez realizada la instalación se procede a la configuración de la base de datos:

- Pantalla de presentación de la herramienta

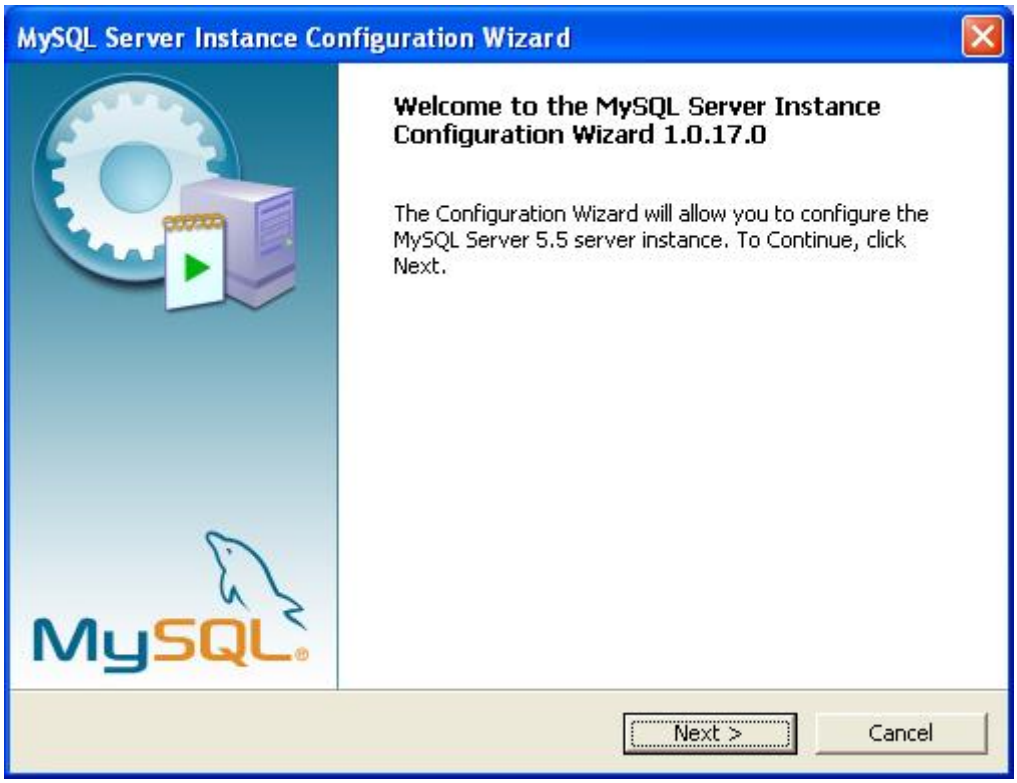


- Segunda panta de presentación de la herramienta

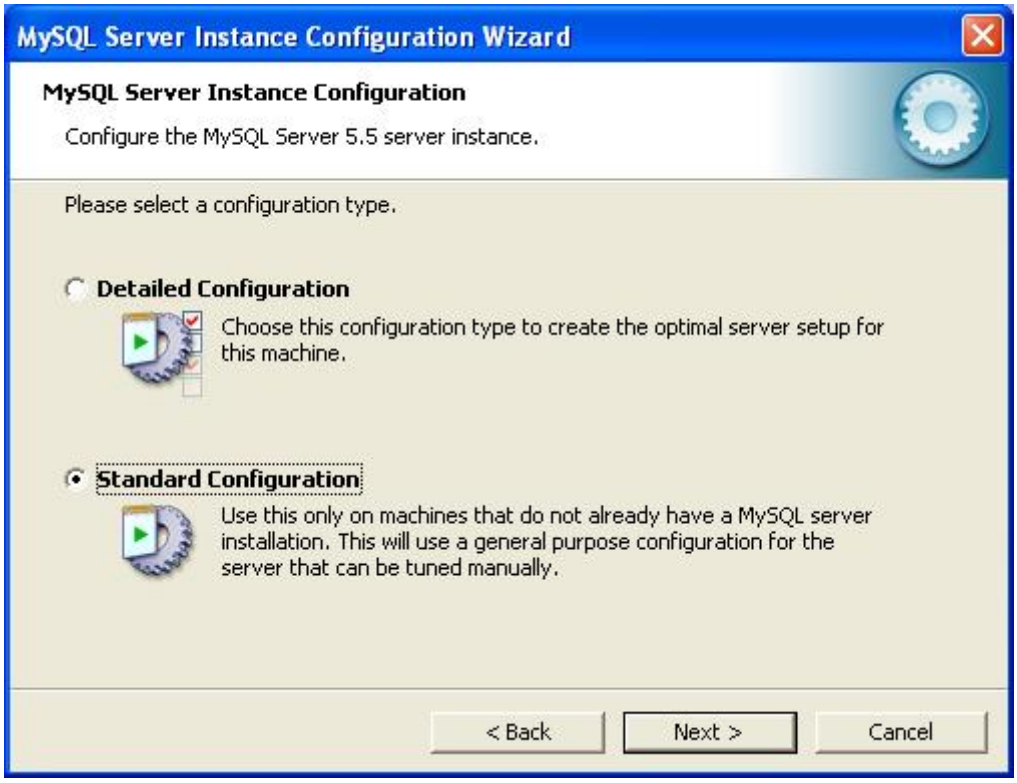
- Tercera pantalla de presentación de la herramienta y lanzamiento de la configuración



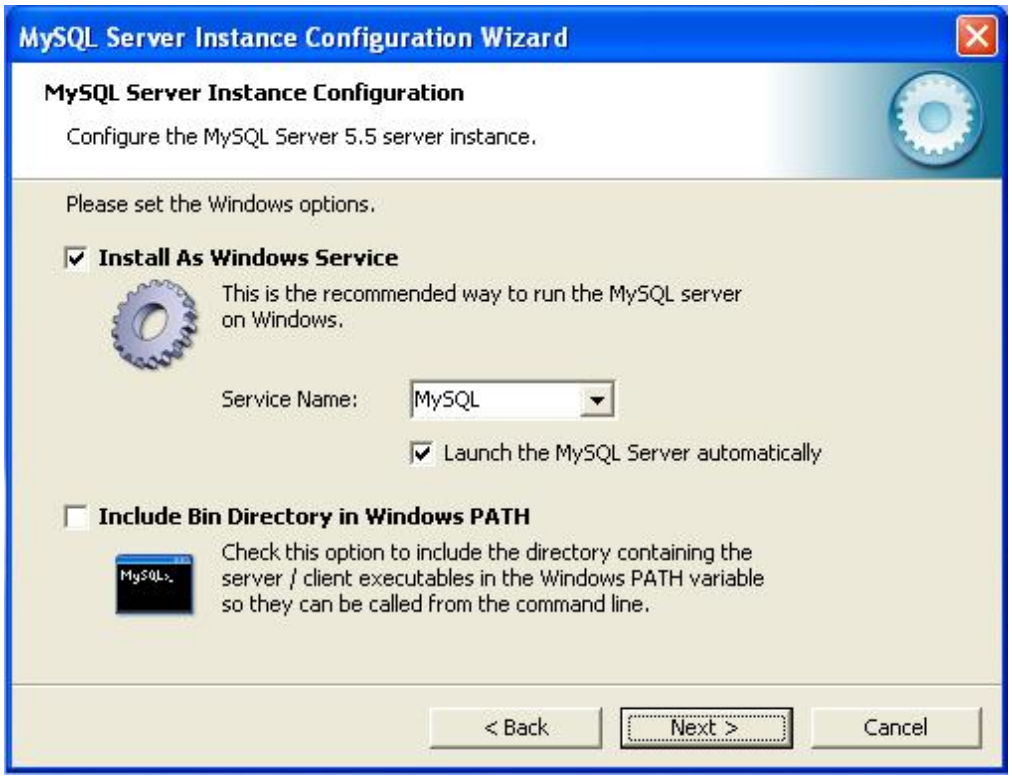
- Pantalla de inicio de la configuración



- Pantalla de tipo de configuración, en este caso se selecciona la estándar.

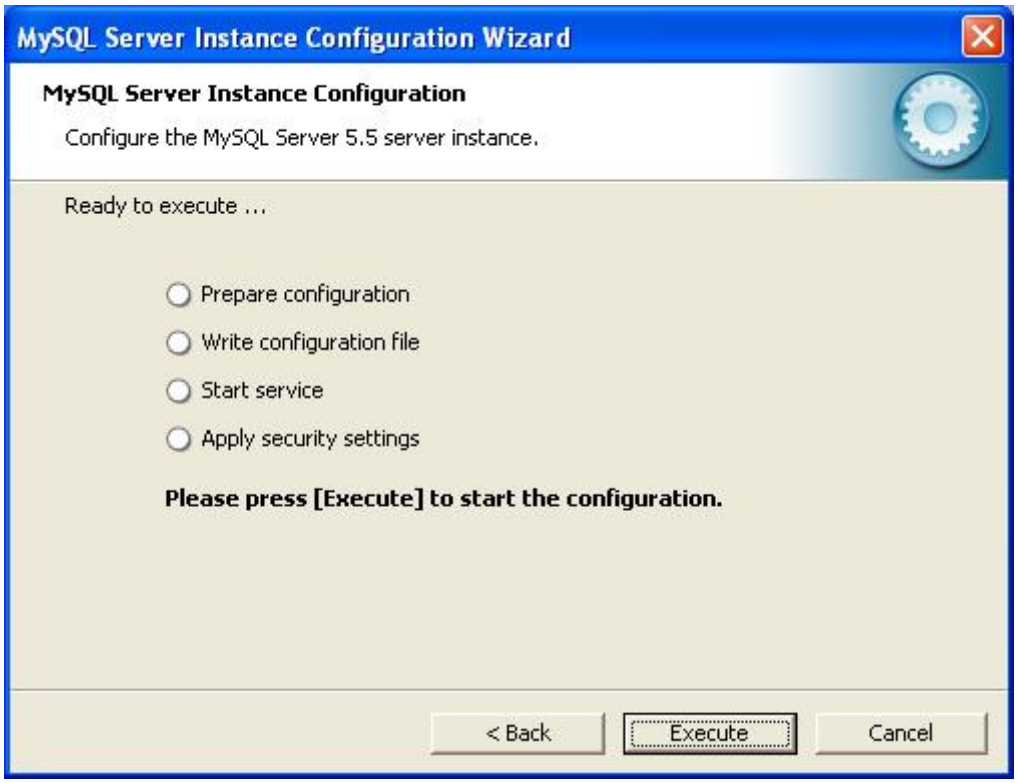


- Pantalla de configuración del servicio, en este caso se indica que se instale como un servicio y con el nombre MySQL. No se incluye el bin en el PATH porque no se le va a dar uso en este proyecto

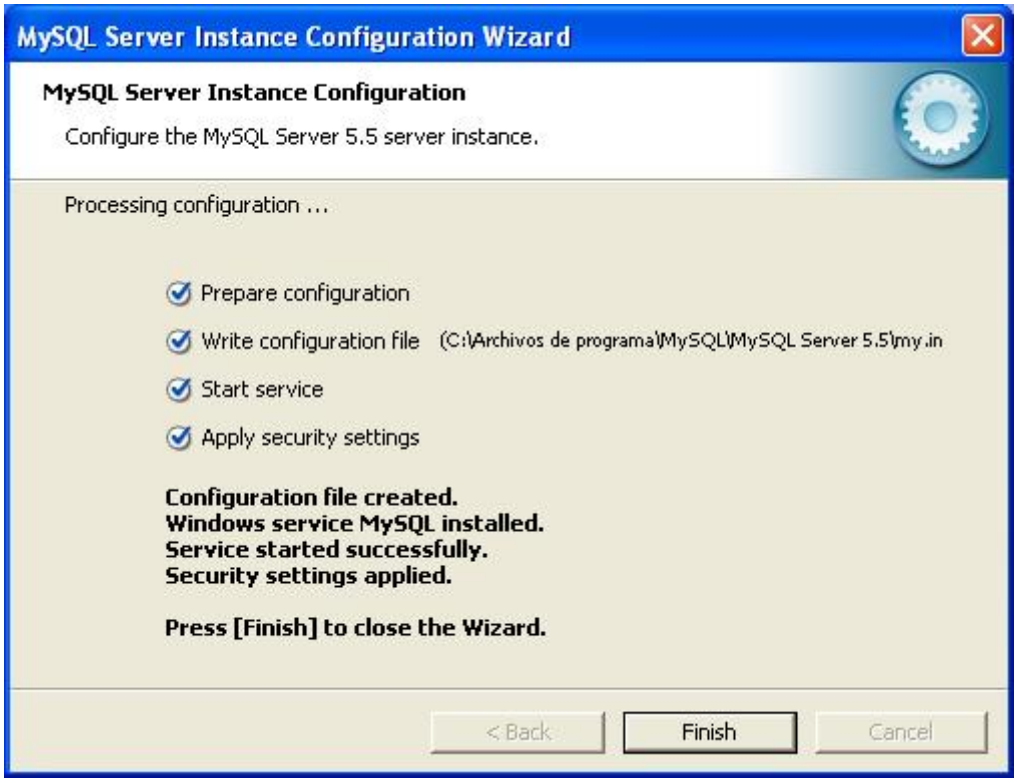


Pantalla de configuración de usuario root. Indicando su contraseña

- Pantalla de pasos de configuración

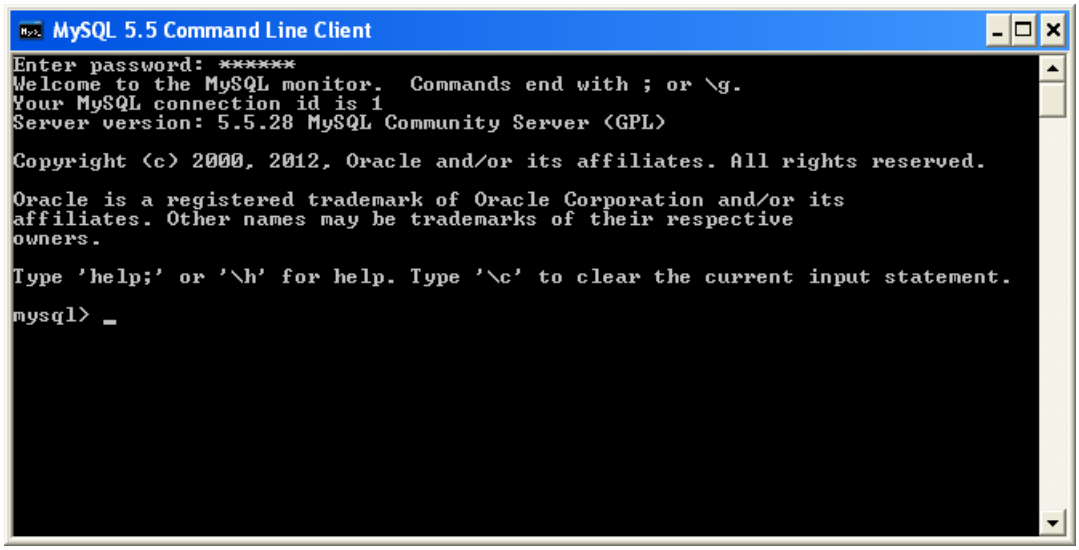


- Pantalla de resultados

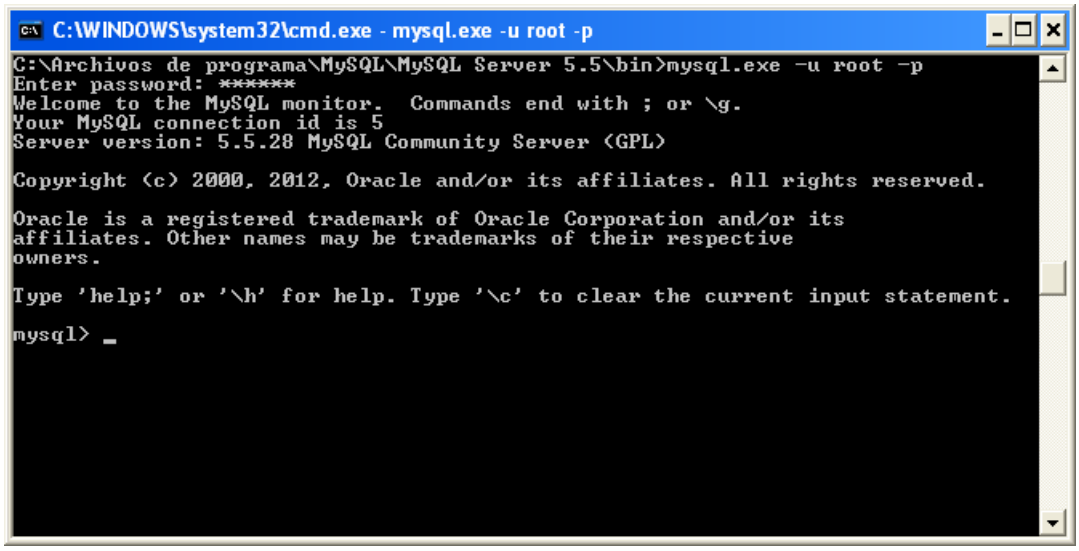


Una vez configurada la herramienta de base de datos se procede a crear la base de datos.

Para ello se accede a la consola de acceso a la base de datos. Se puede acceder desde el menú de inicio > todos los programas > MySQL > MySQL Server 5.5 > MySQL 5.5 Command Line Client e indicar el password de administrador



O accediendo al directorio de instalación de la herramienta de base de datos y ejecutando el comando *"mysql.exe –u root –p"* e indicando el password



continuación crearemos la base de datos donde vamos a realizar nuestras operaciones. Para ello ejecutaremos el comando "CREATE DATABASE PFC". Con ello ya está preparada la basa de datos para cargar los script de creación e inserción de datos de nuestra base de datos. Estos scripts se muestran a continuación:

SCRIPT creación de tablas:

```
CREATE TABLE PERSONA(
        DNI             CHAR(9),
        NOMBRE          VARCHAR2(20)        CONSTRAINT nn_nonbre NOT NULL,
        APELLIDO1       VARCHAR2(20)        CONSTRAINT nn_apellido1 NOT NULL,
        APELLIDO2       VARCHAR2(20),
        CONSTRAINT      pk_persona          PRIMARY KEY (DNI)
);

CREATE TABLE EVENTO(
        CODIGO          NUMBER,
        FECHA           DATE                CONSTRAINT nn_fecha NOT NULL,
        HORA_INI        CHAR(5)             CONSTRAINT nn_hora_ini NOT NULL,
        HORA_FIN        CHAR(5)             CONSTRAINT nn_hora_fin NOT NULL,
        DESCRIPCION     VARCHAR2(200),
        CONSTRAINT      pk_evento           PRIMARY KEY (CODIGO)
);

CREATE TABLE ASISTENTES(
        CODIGO          NUMBER,
        DNI             CHAR(9),
        CONFIRMADO      BOOLEAN         CONSTRAINT nn_confirmado NOT NULL,
        CONSTRAINT      pk_asistentes   PRIMARY KEY (CODIGO, DNI),
        CONSTRAINT      fk1_asistentes  FOREIGN KEY (PERSONA) REFERENCES PERSONA(DNI),
        CONSTRAINT      fk2_asistentes  FOREIGN KEY (EVENTO) REFERENCES EVENTO(CODIGO)
);
```

SCRIPT inserción de datos:

```
INSERT INTO PERSONA VALUES      ('00000001A', 'DIEGO', 'ALCONADA', 'GONZALEZ'),
                                ('00000002A', 'PEDRO', 'GARCIA', 'FERNANDEZ'),
                                ('00000003A', 'RAMON', 'PEREZ', 'GONZALEZ')
                                ('00000004A', 'LORENA', 'VELASCO', NULL),
                                ('00000005A', 'ESTHER', 'GONZALEZ', ''),
                                ('00000006A', 'ALBERTO', 'GARCIA',);

INSERT INTO EVENTO VALUES       (1, '2013-01-01', '12:00', '18:00', 'COMIDA AÑO NUEVO'),
                                (2, '2013-02-01', '13:00', '18:00', 'COMIDA FEBRERO'),
                                (3, '2013-03-01', '14:00', '18:00', 'TEATRO'),
                                (4, '2013-04-01', '18:00', '18:30', 'REUNION COMUNIDAD'),
                                (5, '2013-05-01', '16:00', '18:00', 'PADEL'),
                                (6, '2013-06-01', '17:00', '18:00', 'PARTIDO FUTBOL'),
                                (7, '2013-07-01', '18:00', '19:00', 'PISCINA'),
                                (8, '2013-08-01', '19:00', '20:00', 'CAÑAS'),
                                (9, '2013-09-01', '20:00', '22:00', 'CENA'),
                                (10, '2013-09-02', '21:00', '23:00', 'CENA 2');

INSERT INTO ASISTENTES VALUES   (1,     '00000001A', TRUE),
                                (2,     '00000001A', FALSE),
                                (3,     '00000001A', TRUE),
                                (4,     '00000001A', FALSE),
                                (5,     '00000001A', TRUE),
                                (6,     '00000001A', TRUE),
                                (7,     '00000001A', FALSE),
                                (8,     '00000001A', TRUE),
                                (9,     '00000001A', TRUE),
                                (10,    '00000001A', TRUE),
                                (1,     '00000002A', TRUE),
                                (2,     '00000002A', TRUE),
```

```
(3,     '00000002A', FALSE),
(4,     '00000003A', TRUE),
(5,     '00000003A', TRUE),
(6,     '00000003A', TRUE),
(7,     '00000004A', TRUE),
(8,     '00000004A', FALSE),
(9,     '00000004A', TRUE),
(10,    '00000005A', TRUE),
(1,     '00000006A', TRUE),
(2,     '00000005A', FALSE),
(3,     '00000004A', TRUE);
```

Para cargar los scripts en la base de datos hay que ejecutar los siguientes comandos, desde la carpeta bin de la herramienta de base de datos (C:\Archivos de programa\MySQL\MySQL Server 5.5\bin):

- mysql.exe -u root -pPASS PFC < SCRIPT_CREACION.sql
- mysql.exe -u root -pPASS PFC < SCRIPT_INSERCION.sql

Siendo –u root la indicación del usuario (Administrador) y –pPASS el password

7.1.2 Instalación de JDK

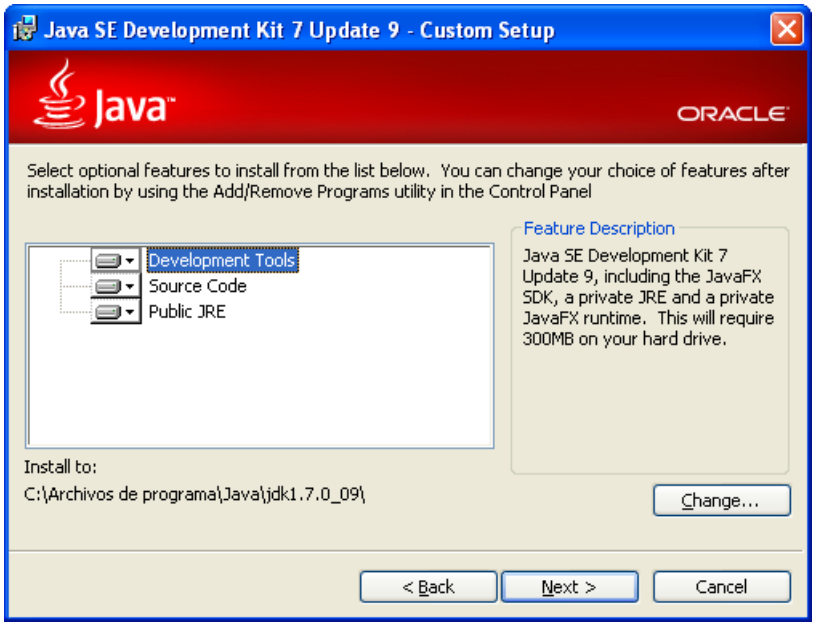
JDK (Java development Kit) es la herramienta de java necesaria para poder desarrollar cualquier api java.

Descargamos y ejecutamos la última versión de la siguiente url:

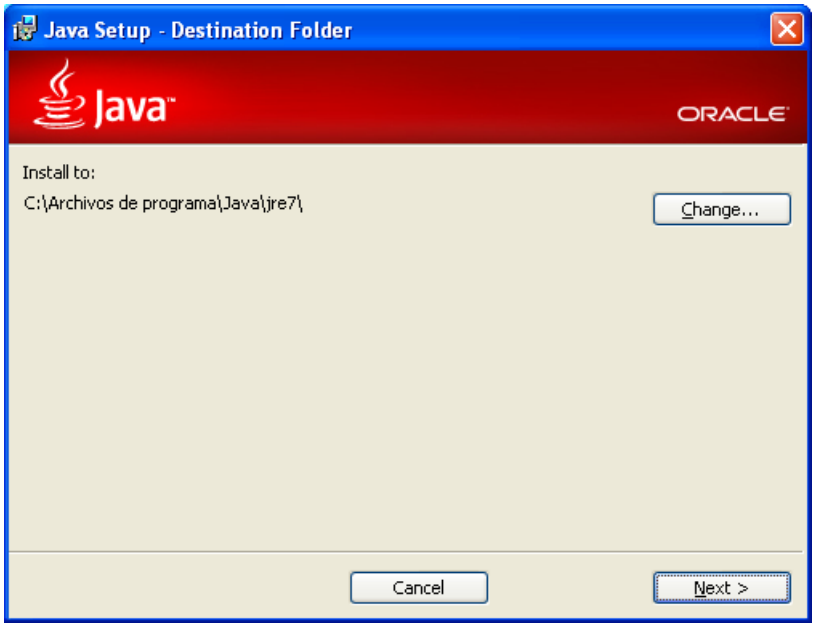
http://www.oracle.com/technetwork/java/javase/downloads/index.html

- Pantalla de inicio de instalación

- Pantalla de configuración de directorio y paquetes a instalar



- Pantalla de instalación de JRE (java runtime environment) necesario para el uso de programas java



- Pantalla de resumen de instalación

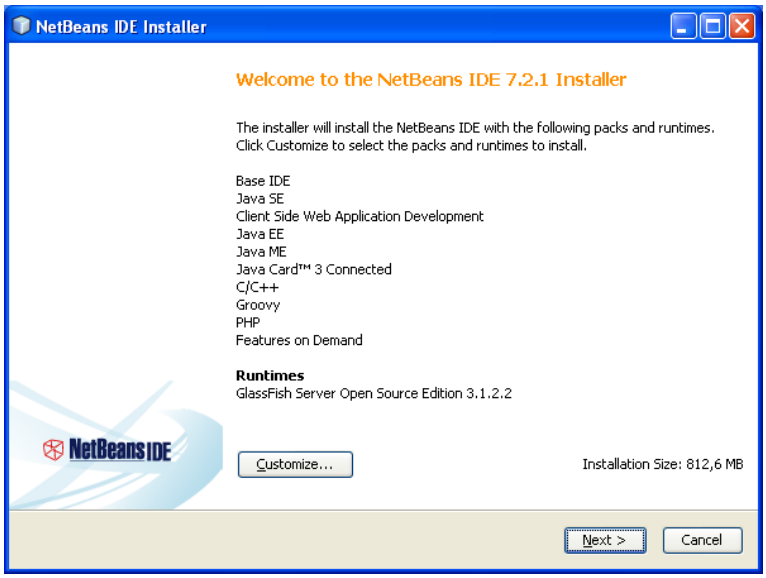
7.1.3 Instalación NetBeans

Herramienta de desarrollo de aplicaciones java. Con ella se desarrollara tanto el entorno gráfico como la herramienta de consulta SQL por medio de lenguaje natural.

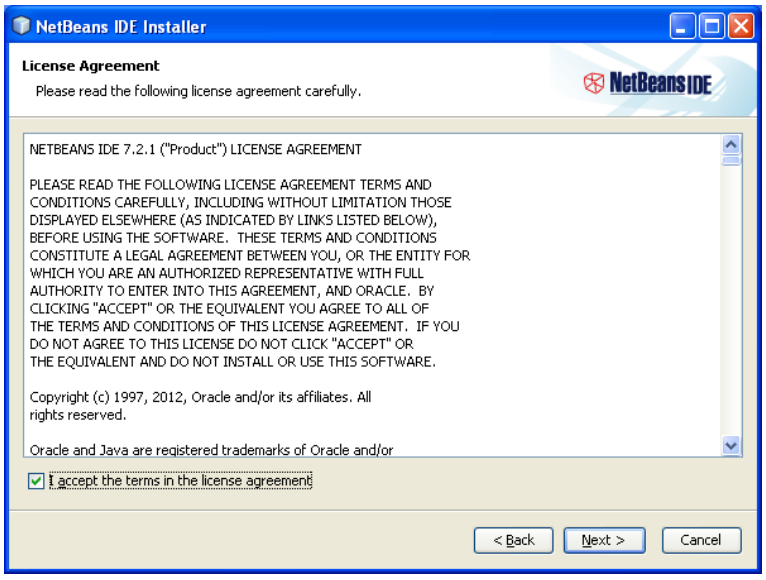
Descargamos y ejecutamos la última versión de la siguiente url:

http://netbeans.org/downloads/

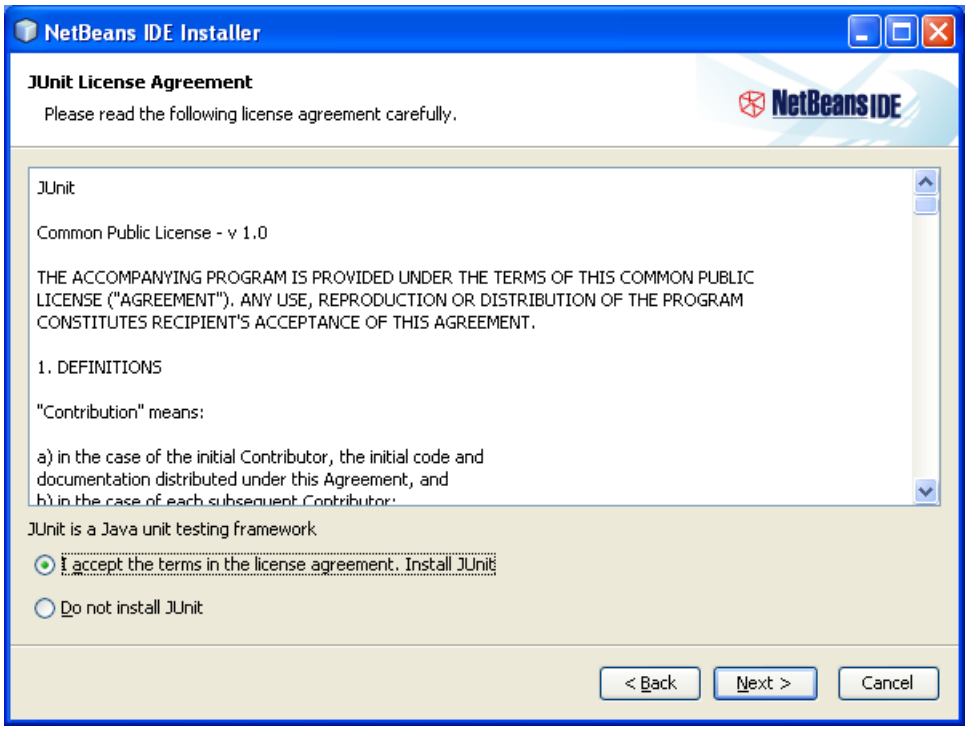
- Pantalla de inicio de instalación



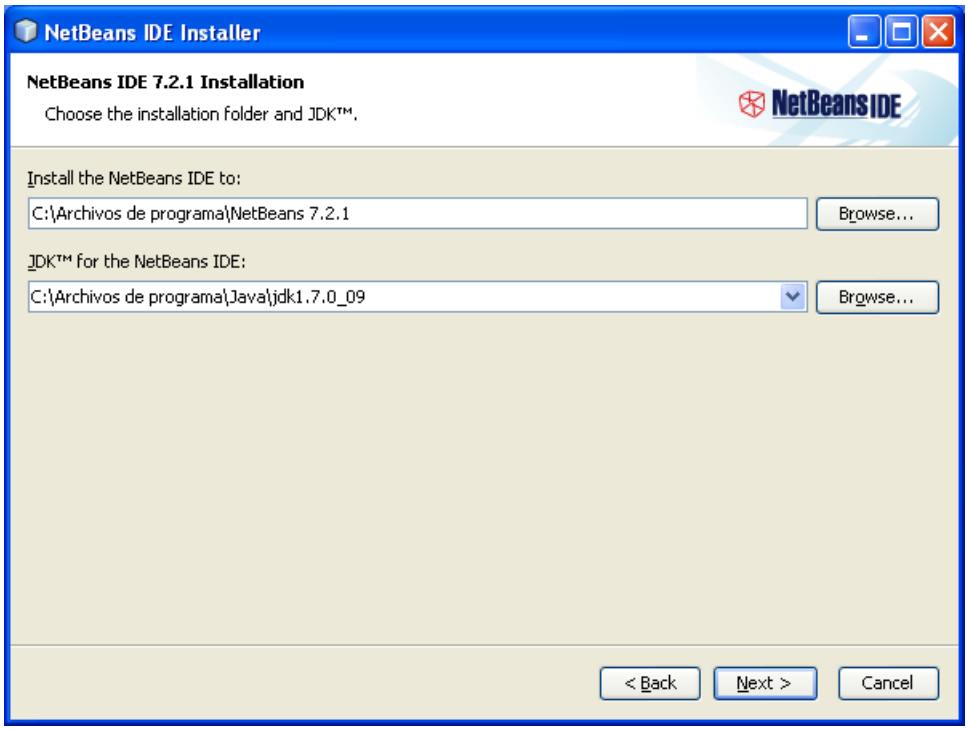
- Pantalla aceptación de licencia



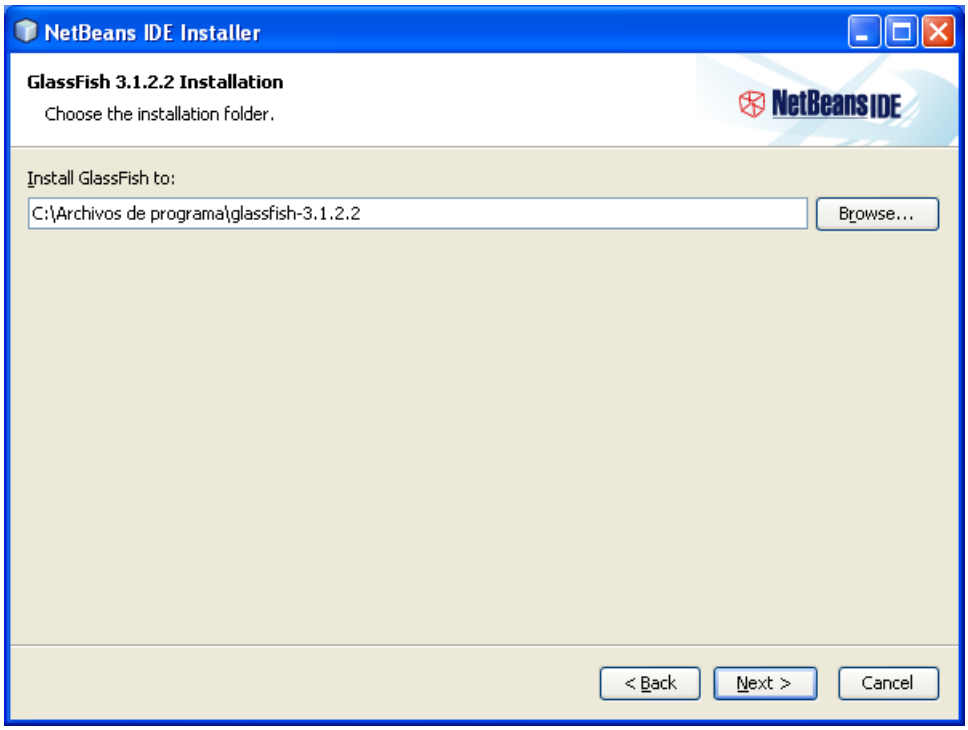
- Pantalla aceptación licencia JUnit



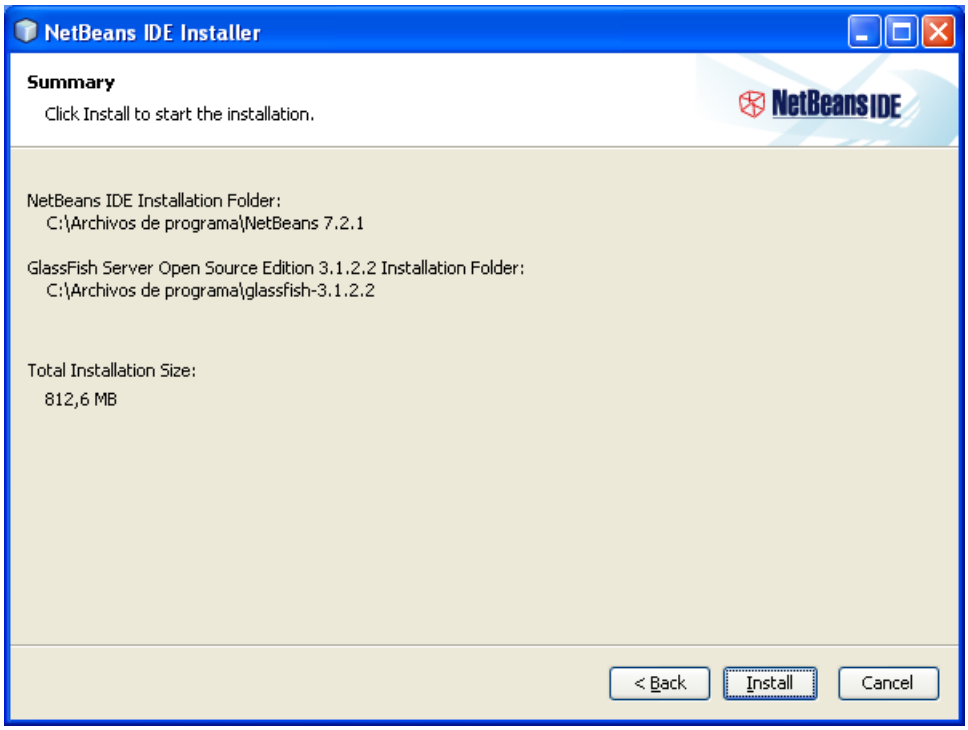
- Pantalla configuración de directorio de instalación y JDK



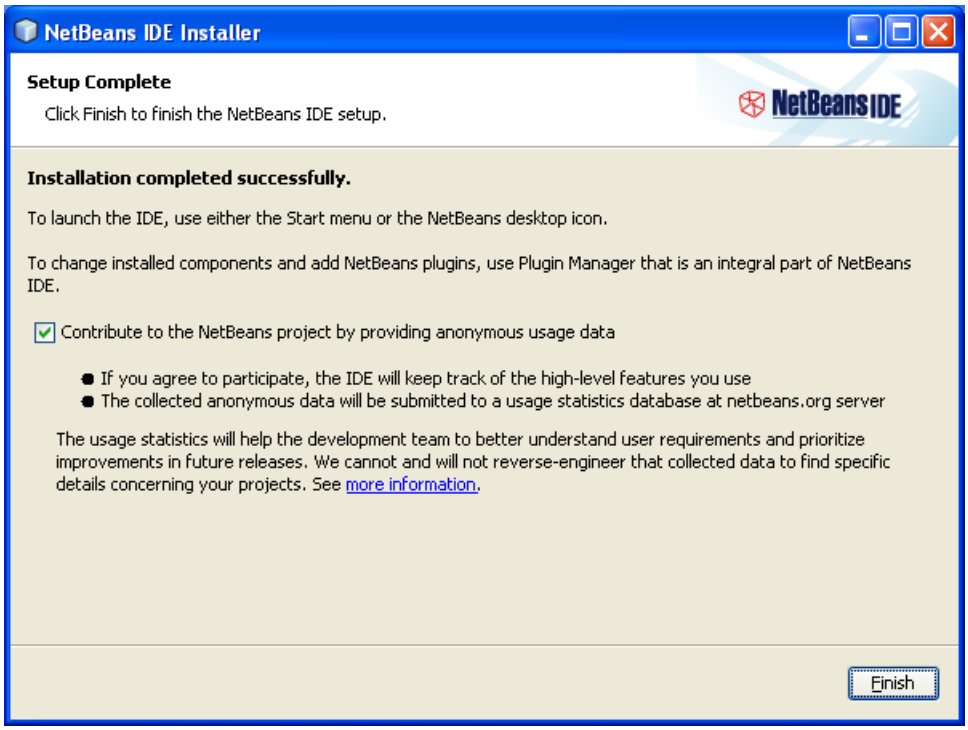
- Pantalla instalación directorio GlassFish



- Pantalla resumen instalación



- Pantalla resultado instalación



7.1.4 Instalación Flex y CUP

- Descargamos JLex de su página web:

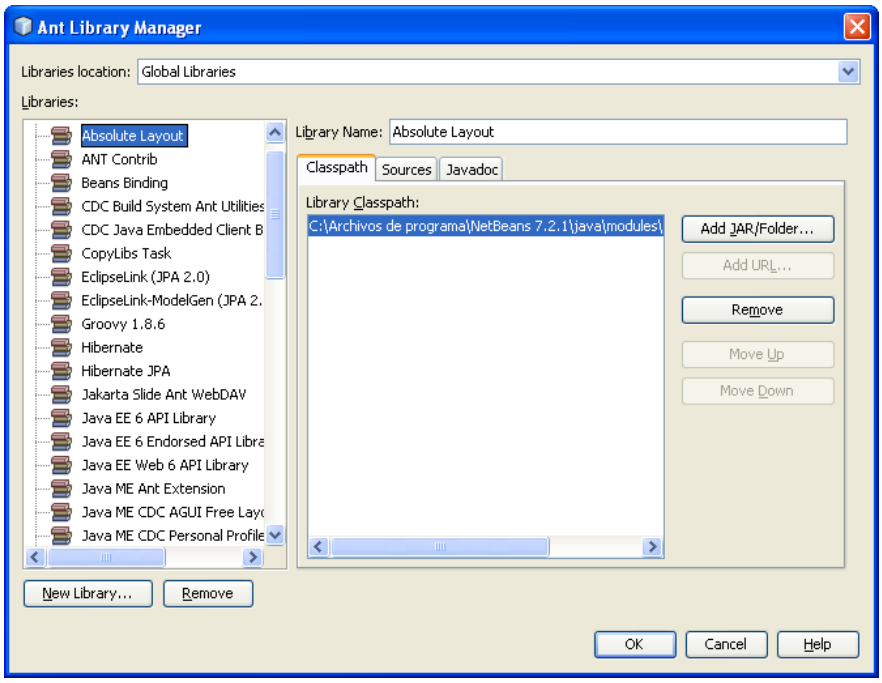
http://jflex.de/download.html

- Descomprimimos el zip descargado es su interior tiene una carpeta lib donde se encuentra el archivo JFlex.jar
- Descargamos Cup de su página web:

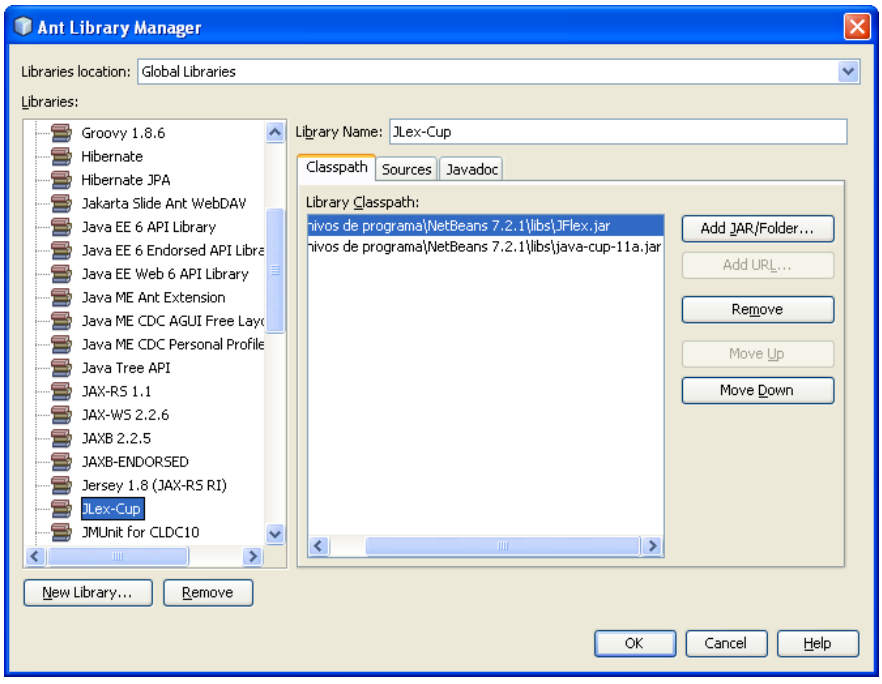
http://www2.cs.tum.edu/projects/cup/

- De la descarga se obtendrá el archivo java-cup-11a.jar

Añadiremos las librerías al repositorio de librerías de Netbeans. Para ello. En la pestaña Tools > Ant Libraries



New library, indicamos el nombre de la librería y añadimos los dos jar que hemos descargado antes.



Por ultimo crearemos un proyecto sencillo para confirmar que funciona

Una vez creado el proyecto, lo primero es añadir la librería antes creada.

Para ello. Botón derecho sobre el proyecto > propiedades y en la categoría librerías se añade

7.2 JLEX

```
package PFC;

import java.io.*;
import java_cup.runtime.Symbol;
import java.lang.String;

%%
%{
        //Definicion de Variables
        public int getChar(){ return yychar; }
%}

%class Lex
%full
%line1
%char
%cup

%state COLUMNAS, WHERE, WHEREVALUE, WHEREBETWEENVALUE, WHEREINVALUE

blank=[\n\r\t ]
number=[0-9]+
ident=[a-z_]+
hora = {number}{number}:{number}{number}
dni={number}{number}{number}{number}{number}{number}{number}{number}[a-z]
fecha={number}{number}{number}{number}\-{number}{number}\-{number}{number}
valor={hora}|{dni}|{fecha}|{number}|{ident}


%%

<YYINITIAL>{blank}+                     { /* Do nothing */ }
<YYINITIAL>mostrar                      { yybegin(COLUMNAS); return new Symbol(sym.TK_SELECT); }
<YYINITIAL>muestra                      { yybegin(COLUMNAS); return new Symbol(sym.TK_SELECT); }
<YYINITIAL>selecciona                   { yybegin(COLUMNAS); return new Symbol(sym.TK_SELECT); }
<YYINITIAL>que                          { yybegin(WHERE); return new Symbol(sym.TK_WHERE); }
<YYINITIAL>de                           { yybegin(WHERE); return new Symbol(sym.TK_WHERE); }
<YYINITIAL>con                          { yybegin(WHERE); return new Symbol(sym.TK_WHERE); }
<YYINITIAL>.                            { parser.addCharNotUsed(yytext(),yychar); }

<COLUMNAS>{blank}+                      { /* Do nothing */ }
<COLUMNAS>tod[oa]s?                     { return new Symbol(sym.TK_ALL); }
<COLUMNAS>dni                           { return new Symbol(sym.TK_DNI); }
<COLUMNAS>nombre                        { return new Symbol(sym.TK_NOMB); }
<COLUMNAS>apellido1?                    { return new Symbol(sym.TK_AP1); }
<COLUMNAS>apellido2                     { return new Symbol(sym.TK_AP2); }
<COLUMNAS>primer\ apellido              { return new Symbol(sym.TK_AP1); }
<COLUMNAS>segundo\ apellido             { return new Symbol(sym.TK_AP2); }

<COLUMNAS>codigo                        { return new Symbol(sym.TK_COD); }
<COLUMNAS>confirmados?                  { return new Symbol(sym.TK_CONFIR); }

<COLUMNAS>fecha                         { return new Symbol(sym.TK_FEC); }
<COLUMNAS>hora\ inicio                  { return new Symbol(sym.TK_H_INI); }
<COLUMNAS>hora\ fin                     { return new Symbol(sym.TK_H_FIN); }
<COLUMNAS>descripcion                   { return new Symbol(sym.TK_DESC); }

<COLUMNAS>personas?                     { yybegin(YYINITIAL); return new Symbol(sym.TK_PERSONA); }
<COLUMNAS>eventos?                      { yybegin(YYINITIAL); return new Symbol(sym.TK_EVENTO); }
<COLUMNAS>asistentes?                   { yybegin(YYINITIAL); return new Symbol(sym.TK_ASISTENTE); }
```

```
<COLUMNAS>.                                                    {
parser.addCharNotUsed(yytext(),yychar); }

<WHERE>{blank}+                          { /* Do nothing */ }

<WHERE>"("                               { return new Symbol(sym.TK_I_PAR); }
<WHERE>")"                               { return new Symbol(sym.TK_D_PAR); }
<WHERE>y                                 { return new Symbol(sym.TK_Y); }
<WHERE>o                                 { return new Symbol(sym.TK_O); }

<WHERE>dni                               { return new Symbol(sym.TK_DNI); }
<WHERE>nombre                            { return new Symbol(sym.TK_NOMB); }
<WHERE>apellido1?                        { return new Symbol(sym.TK_AP1); }
<WHERE>apellido2                         { return new Symbol(sym.TK_AP2); }
<WHERE>primer\ apellido                  { return new Symbol(sym.TK_AP1); }
<WHERE>segundo\ apellido                 { return new Symbol(sym.TK_AP2); }

<WHERE>codigo                            { return new Symbol(sym.TK_COD); }
<WHERE>confirmados?                      { return new Symbol(sym.TK_CONFIR); }

<WHERE>fecha                             { return new Symbol(sym.TK_FEC); }
<WHERE>hora\ inicio                      { return new Symbol(sym.TK_H_INI); }
<WHERE>hora\ fin                         { return new Symbol(sym.TK_H_FIN); }
<WHERE>descripcion                       { return new Symbol(sym.TK_DESC); }

<WHERE>mayor\ que{blank}+                { yybegin(WHEREVALUE); return new Symbol(sym.TK_GT); }
<WHERE>menor\ que{blank}+                { yybegin(WHEREVALUE); return new Symbol(sym.TK_LT); }
<WHERE>mayor\ o\ igual\ que{blank}+      { yybegin(WHEREVALUE); return new Symbol(sym.TK_GET); }
<WHERE>menor\ o\ igual\ que{blank}+      { yybegin(WHEREVALUE); return new Symbol(sym.TK_LET); }
<WHERE>igual(\ a)?{blank}+               { yybegin(WHEREVALUE); return new Symbol(sym.TK_EQ); }
<WHERE>distinto(\ de)?{blank}+           { yybegin(WHEREVALUE); return new Symbol(sym.TK_NEQ); }
<WHERE>entre{blank}+          { yybegin(WHEREBETWEENVALUE); return new Symbol(sym.TK_BETWEEN); }
<WHERE>(incluidos?\ )?en{blank}+         { yybegin(WHEREINVALUE); return new Symbol(sym.TK_IN); }
<WHERE>.                                 { parser.addCharNotUsed(yytext(),yychar);}

<WHEREVALUE>{blank}+                     { yybegin(WHERE); }
<WHEREVALUE>mostrar                      { yybegin(COLUMNAS); return new Symbol(sym.TK_SELECT); }
<WHEREVALUE>muestra                      { yybegin(COLUMNAS); return new Symbol(sym.TK_SELECT); }
<WHEREVALUE>selecciona                   { yybegin(COLUMNAS); return new Symbol(sym.TK_SELECT); }
<WHEREVALUE>")"                          { yybegin(WHERE); return new Symbol(sym.TK_D_PAR); }
<WHEREVALUE>{valor}                      { return new Symbol(sym.TK_VALOR, new String (yytext())); }
<WHEREVALUE>.                            { parser.addCharNotUsed(yytext(),yychar);}

<WHEREBETWEENVALUE>{blank}+              { /* Do nothing */ }
<WHEREBETWEENVALUE>y                     { return new Symbol(sym.TK_Y); }
<WHEREBETWEENVALUE>{valor}               { return new Symbol(sym.TK_VALOR, new String (yytext())); }
<WHEREBETWEENVALUE>.                     { parser.addCharNotUsed(yytext(),yychar);}

<WHEREINVALUE>{blank}+                   { /* Do nothing */ }
<WHEREINVALUE>mostrar                    { yybegin(COLUMNAS); return new Symbol(sym.TK_SELECT); }
<WHEREINVALUE>muestra                    { yybegin(COLUMNAS); return new Symbol(sym.TK_SELECT); }
<WHEREINVALUE>selecciona                 { yybegin(COLUMNAS); return new Symbol(sym.TK_SELECT); }

<WHEREINVALUE>y                          { parser.addCharNotUsed(yytext(),yychar); }
<WHEREINVALUE>o                          { parser.addCharNotUsed(yytext(),yychar); }

<WHEREINVALUE>{valor}                    { return new Symbol(sym.TK_VALOR, new String (yytext())); }
<WHEREINVALUE>.                          { parser.addCharNotUsed(yytext(),yychar);}
```

7.3 CUP

```
package PFC;

import java.io.*;
import java.text.*;
import java.lang.String ;
import java.util.*;
import java.util.concurrent.*;
import java_cup.runtime.*;

parser code {:

	static String strSQL = "";
	static ArrayList<String> wordsNotUsed = null;
	static Integer postLastWordNotUsed = null;

	/* addCharNotUsed function */
	public static void addCharNotUsed (String auxStr, int pos)
	{

		if ( ++postLastWordNotUsed == pos )
		{
			int auxInt = wordsNotUsed.size() - 1;
			String word = wordsNotUsed.remove(auxInt);
			word += auxStr;
			wordsNotUsed.add(word);

		}
		else
		{
			postLastWordNotUsed = pos;
			wordsNotUsed.add(auxStr);
		}

	} /* End addCharNotUsed */

	/* getWordsNotUsed function */
	public static ArrayList<String> getWordsNotUsed ()
	{
		return wordsNotUsed;
	}/* End getWordsNotUsed */

	/* generaSql function */
	public static String generaSql (String auxStr) throws Exception
	{

		//definicion de variables
		Lex lex;
		parser scanner;
		strSQL = "";
		wordsNotUsed = new ArrayList<String>();
		postLastWordNotUsed = -99;

		lex = new Lex (new java.io.StringReader(auxStr.toLowerCase()));
		scanner = new parser(lex);
		scanner.parse();

		return strSQL;

	} /* End generaSql */
```

```
:};


//terminals
terminal            TK_SELECT, TK_ALL, TK_DNI, TK_NOMB, TK_AP1, TK_AP2, TK_COD, TK_CONFIR, TK_FEC, TK_H_INI,
TK_H_FIN, TK_DESC;
terminal            TK_PERSONA, TK_EVENTO, TK_ASISTENTE;
terminal            TK_WHERE, TK_GT, TK_LT, TK_GET, TK_LET, TK_EQ, TK_NEQ, TK_I_PAR, TK_D_PAR, TK_Y, TK_O,
TK_BETWEEN, TK_IN;
terminal String     TK_VALOR;

//non terminals
non terminal                    program;
non terminal Stringselect, colums, colums2, table, where, where2, comp, atrib, whereIn;

//precedence
precedence left TK_Y, TK_O;

//Grammar

program  ::=        select:s1                       {: parser.strSQL = s1 + ";"; :}
         |          error                          {: parser.strSQL = "ERROR: estructura SELECT no reconocida"; :}
         ;

select   ::=        TK_SELECT colums:s1 table:s2            {: RESULT = "SELECT " + s1 + " FROM " + s2; :}
         |          TK_SELECT colums:s1 table:s2 TK_WHERE where:s3
                                                 {: RESULT = "SELECT " + s1 + " FROM " + s2 + " WHERE " + s3; :}
         ;

colums   ::=        colums2:s1                      {: RESULT = s1; :}
         |          TK_ALL                         {: RESULT = "*"; :}
         |                                         {: RESULT = "*"; :}
         ;

colums2  ::=        colums2:s1 atrib:s2             {: RESULT = s1 + ", " + s2; :}
         |          atrib:s1                       {: RESULT = s1; :}
         ;

where    ::=        where2:s1                       {: RESULT = s1; :}
         |          atrib:s1 TK_BETWEEN TK_VALOR:s2 TK_Y TK_VALOR:s3
                                                 {: RESULT = s1 + " BETWEEN '" + s2 + "' AND '" + s3 +"'"; :}
         |          atrib:s1 TK_IN whereIn:s2              {: RESULT = s1 + " IN ( " + s2 + " )"; :}
         |          atrib:s1 TK_IN select:s2               {: RESULT = s1 + " IN ( " + s2 + " )"; :}
                    ;

where2   ::=        TK_I_PAR where2:s1 TK_D_PAR     {: RESULT = "(" + s1 + ")"; :}
         |          where2:s1 TK_Y where2:s2          {: RESULT = s1 + " AND " + s2; :}
         |          where2:s1 TK_O where2:s2          {: RESULT = s1 + " OR " + s2; :}
         |          TK_I_PAR where2:s1 TK_Y where2:s2 TK_D_PAR          {: RESULT = "(" + s1 + " AND " + s2 + ")"; :}
         |          TK_I_PAR where2:s1 TK_O where2:s2 TK_D_PAR          {: RESULT = "(" + s1 + " OR " + s2 + ")"; :}
         |          atrib:s1 comp:s2 TK_VALOR:s3                        {: RESULT = s1 + " " + s2 + " '" + s3 + "'"; :}
         |          atrib:s1 comp:s2 select:s3                          {: RESULT = s1 + " " + s2 + "(" + s3 + ")"; :}
         ;

whereIn  ::=        whereIn:s1 TK_VALOR:s2          {: RESULT = s1 + ", '" + s2 + "'"; :}
         |          TK_VALOR:s1                    {: RESULT =  "'" + s1 + "'"; :}
         ;

table    ::=        TK_PERSONA                      {: RESULT = "PERSONA"; :}
         |          TK_EVENTO                      {: RESULT = "EVENTO"; :}
         |          TK_ASISTENTE                   {: RESULT = "ASISTENTES"; :}
         ;
```

```
comp    ::=     TK_GT                   {: RESULT = ">"; :}
        |       TK_LT                   {: RESULT = "<"; :}
        |       TK_GET                  {: RESULT = ">="; :}
        |       TK_LET                  {: RESULT = "<="; :}
        |       TK_EQ                   {: RESULT = "="; :}
        |       TK_NEQ                  {: RESULT = "<>"; :}
        ;

atrib   ::=     TK_DNI                  {: RESULT = "DNI"; :}
        |       TK_NOMB                 {: RESULT = "NOMBRE"; :}
        |       TK_AP1                  {: RESULT = "APELLIDO1"; :}
        |       TK_AP2                  {: RESULT = "APELLIDO2"; :}
        |       TK_COD                  {: RESULT = "CODIGO"; :}
        |       TK_CONFIR               {: RESULT = "CONFIRMADO"; :}
        |       TK_FEC                  {: RESULT = "FECHA"; :}
        |       TK_H_INI                {: RESULT = "HORA_INI"; :}
        |       TK_H_FIN                {: RESULT = "HORA_FIN"; :}
        |       TK_DESC                 {: RESULT = "DESCRIPCION"; :}
        ;
```

7.4 AlcoNQL

Clase java que representa el parser de la aplicación.

Es una clase estática con lo que no hay que crear una instancia de ella

7.4.1 Métodos

String generaSQL(String nlStr)

Dado un texto de entrada en lenguaje natural devuelve la sentencia SQL equivalente o error en caso de no encontrar coincidencias con las estructuras definidas

ArrayList<String> getWordsNotUsed()

Devuelve un array de string con las palabras que no han sido usadas en el proceso de parseo

7.5 GUIAlcoNQL

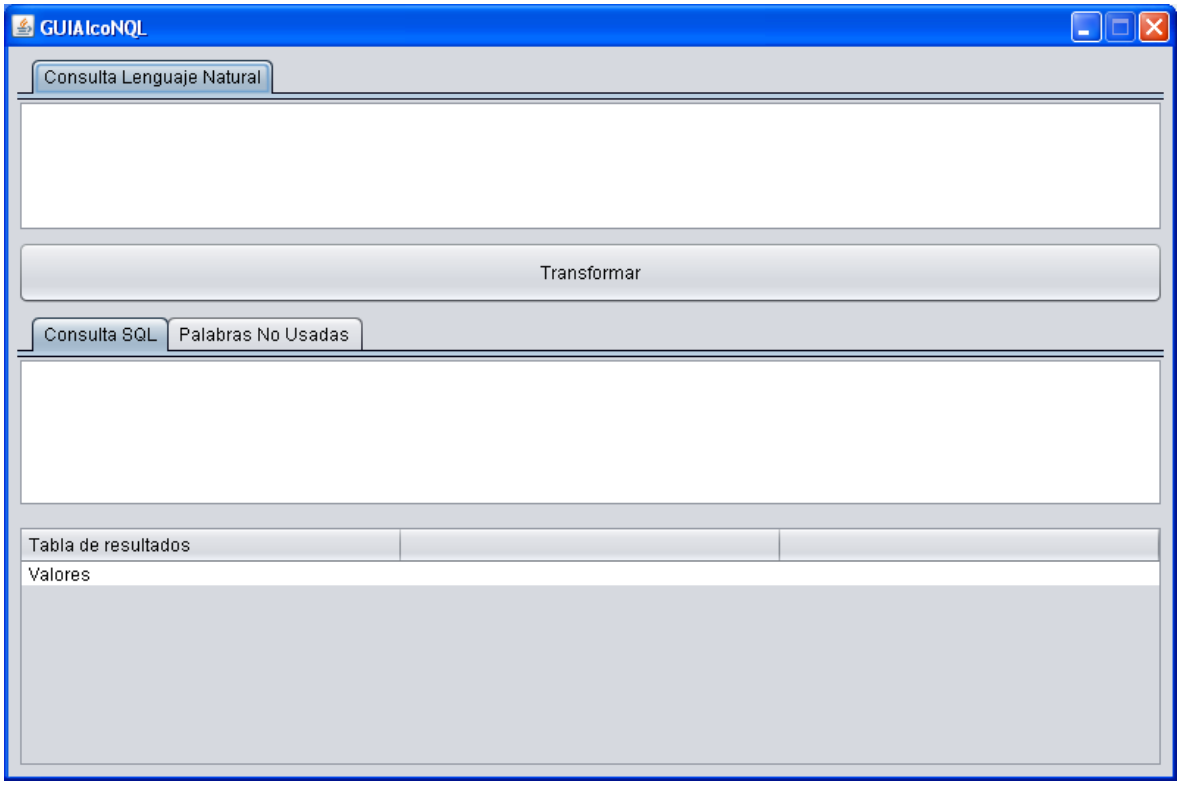
Entorno grafico que hace uso del parseador AlcoNQL para transformar el lenguaje natural en sentencia SQL, realiza la consulta a la base de datos y muestra los resultados

7.5.1 GuiJFrame.java

Entorno visual de la aplicación con:

- Un primer campo de texto donde introducir la consulta en lenguaje natural.
- Un botón de transformación del texto

- Un campo donde se muestra la consulta SQL resultante (Consulta SQL)
- Un campo donde se muestran las palabras no usadas (Palabras no usadas)
- Una tabla donde se muestras los resultados devueltos por la base de datos



7.5.2 NLToSQL.java

Clase que hace uso del parseador AlcoNQL para procesar el lenguaje natural

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package pfc.parser;

import PFC.parser;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author Administrador
 */
public class NLToSQL {

   public NLToSQL(){}

   public String naturalLanguajeToSQL(String nlStr)
   {
      String sqlStr = "";
      try {
```

```
            sqlStr = parser.generaSql(nlStr);
        } catch (Exception ex) {
            Logger.getLogger(NLToSQL.class.getName()).log(Level.SEVERE, null, ex);
        }

        return sqlStr;
    }

    public ArrayList<String> getWordsNotUsed()
    {

        return parser.getWordsNotUsed();
    }

}
```

7.5.3 MySQLQuery.java

Clase que realiza la consulta SQL a la base de datos y obtiene los resultados

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package pfc.bbdd;

import java.sql.*;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author Administrador
 */
public class MySQLQuery {

    //declaracion de variables
    private String bbdd, user, pass;
    private Connection conn = null;

    public MySQLQuery(String bbdd, String user, String pass)
    {
        this.bbdd = bbdd;
        this.user = user;
        this.pass = pass;

            try {
                // The newInstance() call is a work around for some
                // broken Java implementations
                Class.forName("com.mysql.jdbc.Driver").newInstance();
            } catch (InstantiationException | IllegalAccessException | ClassNotFoundException ex) {
                Logger.getLogger(MySQLQuery.class.getName()).log(Level.SEVERE, null, ex);
            }
    }

    public ArrayList<String[]> query(String SQLQuery)
    {
        Statement stmt = null;
        ResultSet rs = null;
```

```
ResultSetMetaData rsMetaData = null;
ArrayList<String[]> tabla = new ArrayList<String[]>();

try
{
   conn = DriverManager.getConnection(bbdd, user, pass);
   stmt = conn.createStatement();
   rs = stmt.executeQuery(SQLQuery);
   rsMetaData = rs.getMetaData();

   int numColums = rsMetaData.getColumnCount();
   String[] columNames = new String[numColums];
   for (int i = 0; i < numColums; i++)
   {
      columNames[i] = rsMetaData.getColumnName(i + 1);
   }
   tabla.add(columNames);

   while(rs.next())
   {
      String[] columI = new String[numColums];
      for (int j = 0; j < numColums; j++)
      {
         columI[j] = rs.getString(j + 1);
      }
      tabla.add(columI);
   }
}
catch (SQLException ex)
{
   String[] cabErr = new String[1];
   String[] err = new String[1];
   cabErr[0] = "Error consulta BBDD";
   tabla.add(cabErr);
   err[0] = ex.getMessage();
   tabla.add(err);
   Logger.getLogger(MySQLQuery.class.getName()).log(Level.SEVERE, null, ex);
}
finally
{
   if (rs != null)
   {
      try
      {
         rs.close();
      }
      catch (SQLException sqlEx) { } // ignore
      rs = null;
   }

   if (stmt != null)
   {
      try
      {
         stmt.close();
      }
      catch (SQLException sqlEx) { } // ignore
      stmt = null;
   }

   if (conn != null)
   {
      try
```

```
            {
               conn.close();
            }
            catch (SQLException sqlEx) { } // ignore
            conn = null;
         }

      }

      return tabla;
   }
}
```

7.6 Instalador

Se ha creado un instalador con la herramienta NSIS. Con él se instalan los siguientes productos:

- GUIAlcoNQL (Herramienta de consulta SQL por medio del lenguaje natural)
- jre-7u9-windows-i586 (Máquina virtual JAVA)
- mysql-5.5.28-win32 (Base de datos MySQL)

7.6.1 Script de instalación

```
; Taken from http://nsis.sourceforge.net/Simple_installer_with_JRE_check by weebib
; Use it as you desire.

; Credit given to so many people of the NSIS forum.

!define AppName "AlcoNQL"
!define AppVersion "2.0"
!define ShortName "AlcoNQL"
!define JRE_VERSION "1.7.09"
!define Vendor "U.O.C."

!define MySQLInstPath "$PROGRAMFILES\MySQL Server 5.5"

!include MUI.nsh
!include Sections.nsh


Var InstallJRE
Var JREPath


;--------------------------------
;Configuration

	;General
	Name "${AppName}"
	OutFile "setup.exe"

	;Folder selection page
	InstallDir "$PROGRAMFILES\${SHORTNAME}"

	;Get install folder from registry if available
```

```
        InstallDirRegKey HKLM "SOFTWARE\${ShortName}" ""

        ; Installation types
        InstType "Cliente"

;--------------------------------
;Pages

  ; License page
  !insertmacro MUI_PAGE_LICENSE "licencia.txt"
  ; This page checks for JRE. It displays a dialog based on JRE.ini if it needs to install JRE
  ; Otherwise you won't see it.
  Page custom CheckInstalledJRE

  ; Define headers for the 'Java installation successfully' page
  !define MUI_INSTFILESPAGE_FINISHHEADER_TEXT "Instalación finalizada"
  !define MUI_PAGE_HEADER_TEXT "Instalación AlcoNQL"
  !define MUI_PAGE_HEADER_SUBTEXT "Bienvenido a la instalación de la aplicación AlcoNQL"
  ;!define MUI_PAGE_HEADER_SUBTEXT "Espere mientras se compelta la instalación de la aplicación"
  !define MUI_INSTFILESPAGE_FINISHHEADER_SUBTEXT "Aplicación instalada correctamente"
  !insertmacro MUI_PAGE_DIRECTORY
  !insertmacro MUI_PAGE_COMPONENTS
  !insertmacro MUI_PAGE_INSTFILES
  !define MUI_INSTFILESPAGE_FINISHHEADER_TEXT "Instalación completada"
  !define MUI_PAGE_HEADER_TEXT "Instalando"
  !define MUI_PAGE_HEADER_SUBTEXT "Por favor, espera mientras ${AppName} se instala."
  !insertmacro MUI_PAGE_FINISH
  !insertmacro MUI_UNPAGE_CONFIRM
  !insertmacro MUI_UNPAGE_INSTFILES

;--------------------------------
;Modern UI Configuration

  !define MUI_ABORTWARNING

;--------------------------------
;Languages

  !insertmacro MUI_LANGUAGE "Spanish"

;--------------------------------
;Language Strings

  ;Header
  LangString TEXT_JRE_TITLE ${LANG_SPANISH} "Java Runtime Environment"
  LangString TEXT_JRE_SUBTITLE ${LANG_SPANISH} "Instalación"
  LangString TEXT_PRODVER_TITLE ${LANG_SPANISH} "Versión instalada de ${AppName}"
  LangString TEXT_PRODVER_SUBTITLE ${LANG_SPANISH} "Instalación cancelada"

;--------------------------------
  ;Only useful for BZIP2 compression

  ReserveFile "jre.ini"
  !insertmacro MUI_RESERVEFILE_INSTALLOPTIONS

;--------------------------------
;Installer Sections

Section -"JRE" jre
        SectionIn 1 2 RO
        Push $0
        Push $1
        ;MessageBox MB_OK "Seccion JRE"
```

```
	Strcmp $InstallJRE "YES" InstallJRE JREPathStorage
	;MessageBox MB_OK "Iniciando la instalación del JRE 1.6 u14"
	DetailPrint "Iniciando la instalación del JRE 1.7 u9"
	InstallJRE:
		;MessageBox MB_OK "Instalando JRE"
		DetailPrint "Iniciando la instalación del JRE"
		File /oname=$TEMP\jre7u9.exe jre-7u9-windows-i586.exe
		ExecWait '"$TEMP\jre7u9.exe" /s /v "/qn REBOOT=Suppress JAVAUPDATE=0 WEBSTARTICON=0"' $0
		Delete $TEMP\jre7u9.exe
		StrCmp $0 "0" InstallVerif 0
		Push "La instalación el JRE ha finalizado de forma anormal."
		Goto ExitInstallJRE

	InstallVerif:
		DetailPrint "Comprobando la instalación del JRE"
		Push "${JRE_VERSION}"
		Call DetectJRE
		Pop $0      ; DetectJRE's return value
		StrCmp $0 "0" ExitInstallJRE 0
		StrCmp $0 "-1" ExitInstallJRE 0
		Goto JavaExeVerif
		Push "La instalación del JRE ha fallado"
		Goto ExitInstallJRE

	JavaExeVerif:
		IfFileExists $0 JREPathStorage 0
		Push "El archivo : $0, no se ha podido localizar."
		Goto ExitInstallJRE

	JREPathStorage:
		;MessageBox MB_OK "JRE Path Storage"
		!insertmacro MUI_INSTALLOPTIONS_WRITE "jre.ini" "UserDefinedSection" "JREPath" $1
		StrCpy $JREPath $0
		Goto End

	ExitInstallJRE:
		Pop $1
		MessageBox MB_OK "The setup is about to be interrupted for the following reason : $1"
		Pop $1    ; Restore $1
		Pop $0    ; Restore $0
		Abort
	End:
		Pop $1    ; Restore $1
		Pop $0    ; Restore $0
	AddSize 96256
SectionEnd

Section "Archivos de programa" principal
	SectionIn 1 RO 2
	SetOutPath $INSTDIR
	File "GUIAlcoNQL.jar"
	SetOutPath "$INSTDIR\lib"
	File /r "lib\"
	;If you need the path to JRE, you can either get it here for from $JREPath
	;!insertmacro MUI_INSTALLOPTIONS_READ $0 "jre.ini" "UserDefinedSection" "JREPath"
	;MessageBox MB_OK "JRE Read: $0"
	;Store install folder
	WriteRegStr HKLM "SOFTWARE\${Vendor}\${ShortName}" "" $INSTDIR

	WriteRegStr HKLM "Software\Microsoft\Windows\CurrentVersion\Uninstall\${ShortName}" "DisplayName" "${AppName}"
	WriteRegStr HKLM "Software\Microsoft\Windows\CurrentVersion\Uninstall\${ShortName}"
```

```
"UninstallString" '"$INSTDIR\uninstall.exe"'
        WriteRegDWORD HKLM "Software\Microsoft\Windows\CurrentVersion\Uninstall\${ShortName}"
"NoModify" "1"
        WriteRegDWORD HKLM "Software\Microsoft\Windows\CurrentVersion\Uninstall\${ShortName}"
"NoRepair" "1"

        CreateDirectory "$SMPROGRAMS\${ShortName}"
        CreateShortCut "$SMPROGRAMS\${ShortName}\Desinstalar.lnk" "$INSTDIR\uninstall.exe" ""
"$INSTDIR\uninstall.exe" 0
        CreateShortCut "$SMPROGRAMS\${ShortName}\${ShortName}.lnk" "$INSTDIR\GUIAlcoNQL.jar" ""
"$INSTDIR\GUIAlcoNQL.jar" 0

        ;Create uninstaller
        WriteUninstaller "$INSTDIR\Uninstall.exe"
        AddSize 4096
SectionEnd

Section "Base de datos" bd
        SectionIn 1 RO 2
        File /oname=$TEMP\mysql55.exe mysql-5.5.28-win32.msi
        DetailPrint "Iniciando la instalación de MySQL Server 5"
        ExecWait 'msiexec /i "$TEMP\mysql55.exe" /quiet INSTALLDIR="${MySQLInstPath}"
DATADIR="${MySQLInstPath}"' $0
        Delete $TEMP\mysql55.exe

        IfFileExists "${MySQLInstPath}\bin\MySQLInstanceConfig.exe" CompletaMySQL ErrorMySQL
        CompletaMySQL:
                DetailPrint "Configurando la instalación de MySQL Server 5"

                ExecWait '"${MySQLInstPath}\bin\MySQLInstanceConfig.exe" -i -q "-
l${MySQLInstPath}\mysql_install_log.txt" "-t${MySQLInstPath}\my-small.ini" \
                                        ServiceName=mysql RootPassword=alconql ServerType=SERVER
DatabaseType=MIXED Port=3306 AddBinToPath=yes StrictMode=yes\
                                        -p${MySQLInstPath} -v5.5.28 Charset=utf8'
                Sleep 1000
          Goto CheckService
        CheckService:
                DetailPrint "Comprobando si el servicio MySQL se encuentra en ejecución"
                SimpleSC::ServiceIsRunning  "mysql"
                Pop $0
                Pop $1
                ;MessageBox MB_OK "Servicio Arrancado $0 - $1"
                StrCmp $1 "1" MySQLServiceOk ErrorMySQL
        MySQLServiceOk:
                DetailPrint "Creando la base de datos y las tablas"
                File /oname=$TEMP\create_BBDD.sql SCRIPT_CREACION_BBDD.sql
                ExecWait 'cmd /C "${MySQLInstPath}\bin\mysql.exe" -u root -palconql < $TEMP\create_BBDD.sql >
log.out' $0
                Delete $TEMP\create_BBDD.sql
                File /oname=$TEMP\create.sql SCRIPT_CREACION.sql
                ExecWait 'cmd /C "${MySQLInstPath}\bin\mysql.exe" -u root -palconql pfc < $TEMP\create.sql >
log.out' $0
                Delete $TEMP\create.sql
                File /oname=$TEMP\pfc.sql SCRIPT_INSERCION.sql
                ExecWait 'cmd /C "${MySQLInstPath}\bin\mysql.exe" -u root -palconql pfc < $TEMP\pfc.sql >
log.out' $0
                Delete $TEMP\pfc.sql
                Goto Fin
        ErrorMySQL:
                MessageBox MB_OK "Ocurrió un error durante la instalación de MySQL 5.1, la instalación no puede
continuar"
                Abort
        Fin:
```

```
	AddSize 55436
SectionEnd

;--------------------------------
;Descripciones
LangString DESC_SecAppFiles ${LANG_SPANISH} "Archivos principales de la aplicación"
LangString DESC_BaseDeDatos ${LANG_SPANISH} "Instalar una base de datos en este ordenador que permita
almacenar la información necesaria para ejecutar la aplicación"

!insertmacro MUI_FUNCTION_DESCRIPTION_BEGIN
	!insertmacro MUI_DESCRIPTION_TEXT ${principal} $(DESC_SecAppFiles)
	!insertmacro MUI_DESCRIPTION_TEXT ${bd} $(DESC_BaseDeDatos)
!insertmacro MUI_FUNCTION_DESCRIPTION_END

;--------------------------------
Function .onInit
	;Extract InstallOptions INI Files
	!insertmacro MUI_INSTALLOPTIONS_EXTRACT "jre.ini"
	;Call SetupSections
FunctionEnd

Function CheckInstalledJRE
	;MessageBox MB_OK "Checking Installed JRE Version"
	Push "${JRE_VERSION}"
	Call DetectJRE
	;Messagebox MB_OK "Done checking JRE version"
	Exch $0   ; Get return value from stack
	StrCmp $0 "0" NoFound
	StrCmp $0 "-1" FoundOld
	Goto JREAlreadyInstalled

	FoundOld:
		;MessageBox MB_OK "Versión del JRE instalada antigüa"
		DetailPrint "Versión del JRE instalada antigüa"
		!insertmacro MUI_INSTALLOPTIONS_WRITE "jre.ini" "Field 1" "Text" "${AppName} requires a more
recent version of the Java Runtime Environment than the one found on your computer. The installation of JRE
${JRE_VERSION} will start."
		!insertmacro MUI_HEADER_TEXT "$(TEXT_JRE_TITLE)" "$(TEXT_JRE_SUBTITLE)"
		!insertmacro MUI_INSTALLOPTIONS_DISPLAY_RETURN "jre.ini"
		Goto MustInstallJRE

	NoFound:
		;MessageBox MB_OK "No se ha encontrado una instalación previa del JRE"
		DetailPrint "No se ha encontrado una instalación previa del JRE"
		!insertmacro MUI_INSTALLOPTIONS_WRITE "jre.ini" "Field 1" "Text" "No se ha econtrado ninguna
versión instalada del Java Runtime Environment en su \

                                                ordenador. La instalación del JRE ${JRE_VERSION} se llevará a cabo de
forma automática."
		!insertmacro MUI_HEADER_TEXT "$(TEXT_JRE_TITLE)" "$(TEXT_JRE_SUBTITLE)"
		!insertmacro MUI_INSTALLOPTIONS_DISPLAY_RETURN "jre.ini"
		Goto MustInstallJRE

	MustInstallJRE:
		Exch $0   ; $0 now has the installoptions page return value
		;Do something with return value here
		;Do something with return value here
		Pop $0    ; Restore $0
		;MessageBox MB_OK "Copiamos YES"
		StrCpy $InstallJRE "YES"
		Return

	JREAlreadyInstalled:
```

```
                        ;MessageBox MB_OK "JRE previamente instalado"
                        DetailPrint "JRE ya instalado"
                        StrCpy $InstallJRE "no"
                        !insertmacro MUI_INSTALLOPTIONS_WRITE "jre.ini" "UserDefinedSection" "JREPath" $JREPATH
                        Pop $0                  ; Restore $0
                        Return

FunctionEnd

Function DetectJRE
  Exch $0  ; Get version requested
                        ; Now the previous value of $0 is on the stack, and the asked for version of JDK is in $0
  Push $1 ; $1 = Java version string (ie 1.5.0)
  Push $2 ; $2 = Javahome
  Push $3 ; $3 and $4 are used for checking the major/minor version of java
  Push $4
  ;MessageBox MB_OK "Detecting JRE"
  ReadRegStr $1 HKLM "SOFTWARE\JavaSoft\Java Runtime Environment" "CurrentVersion"
  ;MessageBox MB_OK "Read : $1"
  StrCmp $1 "" DetectTry2
  ReadRegStr $2 HKLM "SOFTWARE\JavaSoft\Java Runtime Environment\$1" "JavaHome"
  ;MessageBox MB_OK "Read 3: $2"
  StrCmp $2 "" DetectTry2
  Goto GetJRE

DetectTry2:
  ReadRegStr $1 HKLM "SOFTWARE\JavaSoft\Java Development Kit" "CurrentVersion"
  ;MessageBox MB_OK "Detect Read : $1"
  StrCmp $1 "" NoFound
  ReadRegStr $2 HKLM "SOFTWARE\JavaSoft\Java Development Kit\$1" "JavaHome"
  ;MessageBox MB_OK "Detect Read 3: $2"
  StrCmp $2 "" NoFound

GetJRE:
  ;$0 = version requested. $1 = version found. $2 = javaHome
  ;MessageBox MB_OK "Getting JRE"
  IfFileExists "$2\bin\java.exe" 0 NoFound
  StrCpy $3 $0 1                        ; Get major version. Example: $1 = 1.5.0, now $3 = 1
  StrCpy $4 $1 1                        ; $3 = major version requested, $4 = major version found
  ;MessageBox MB_OK "Want $3 , found $4"
  IntCmp $4 $3 0 FoundOld FoundNew
  StrCpy $3 $0 1 2
  StrCpy $4 $1 1 2                      ; Same as above. $3 is minor version requested, $4 is minor version installed
  ;MessageBox MB_OK "Want $3 , found $4"
  IntCmp $4 $3 FoundNew FoundOld FoundNew

NoFound:
  ;MessageBox MB_OK "JRE no encontrado"
  Push "0"
  Goto DetectJREEnd

FoundOld:
  ;MessageBox MB_OK "JRE too old: $3 is older than $4"
  ;Push ${TEMP2}
  Push "-1"
  Goto DetectJREEnd
FoundNew:
  ;MessageBox MB_OK "JRE is new: $3 is newer than $4"

  Push "$2\bin\java.exe"
;  Push "OK"
;  Return
   Goto DetectJREEnd
```

```
DetectJREEnd:
        ; Top of stack is return value, then r4,r3,r2,r1
        Exch    ; => r4,rv,r3,r2,r1,r0
        Pop $4  ; => rv,r3,r2,r1r,r0
        Exch    ; => r3,rv,r2,r1,r0
        Pop $3  ; => rv,r2,r1,r0
        Exch    ; => r2,rv,r1,r0
        Pop $2  ; => rv,r1,r0
        Exch    ; => r1,rv,r0
        Pop $1  ; => rv,r0
        Exch    ; => r0,rv
        Pop $0  ; => rv
FunctionEnd

;--------------------------------
;Uninstaller Section

Section "Uninstall"

  ; remove registry keys
  DeleteRegKey HKLM "SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\${ShortName}"
  DeleteRegKey HKLM "SOFTWARE\${AppName}"
  ; remove shortcuts, if any.
  Delete "$SMPROGRAMS\${ShortName}\*.*"
  Delete "$SMPROGRAMS\${ShortName}"
  ; remove files
  RMDir /r "$INSTDIR\lib"
  RMDir /r "$INSTDIR"

SectionEnd
```