4.- M3M

4.1.- Descripción

M3M es una aplicación pensada para la gestión de proyectos basados en métrica v3. Además de esto, presenta una parte de formación que busca servir de manual de referencia sobre métrica v3 de los usuarios de la aplicación, participantes en proyectos que se gestionen con M3M, y que a su vez sirve como módulo de aprendizaje de métrica v3 para personas interesadas.

Para el almacenamiento de toda la información se desarrolla una base de datos que contiene toda la información referente a métrica v3, así como también toda la información referente a los proyectos que se gestionan mediante la aplicación.

En la aplicación se distinguen tres tipos de usuarios, de forma que cada uno de estos verá la aplicación de una manera distinta, ya que la funcionalidad que esta presenta para ellos es también distinta. Los tres grupos de usuarios son:

- root: este grupo de usuarios es realmente un único usuario, el administrador de la aplicación. El administrador será el encargado de gestionar la herramienta: será el encargado de introducir toda la información, tanto la referente a los proyectos, como aquella información que varíe de métrica v3, será el encargado de la creación del resto de usuarios...
- Participante: este grupo de usuarios estará constituido por aquellas personas que participen en alguno de los proyectos gestionados por la aplicación. M3M permitirá a estos usuarios ver la información relacionada con los proyectos en los que participa: otros usuarios que participen y algunos de sus datos, su trabajo, mensajería... y además, les permitirá consultar la información sobre métrica v3 para resolver sus dudas.
- Invitados: este grupo de usuarios estará constituido por aquellos usuarios que accedan a la aplicación para aprender métrica v3. A estos usuarios la aplicación les muestra toda la información sobre métrica v3 estructurada según a varios criterios.

Para acceder a la aplicación, los pasos a seguir por un usuario sería los siguientes:

- 1. Nuestra aplicación residirá en un servidor Web de forma que sea accesible a través de cualquier ordenador con conexión a Internet. Esto hará que gracias a las nuevas tecnologías podamos acceder a ella desde prácticamente cualquier sitio. Una vez accedamos a la aplicación se nos mostrará la vista de usuario invitado, de forma que este usuario no necesita estar registrado, y además se nos dará la posibilidad de identificarnos, si somos parte de otro grupo de usuarios, para que se nos muestra la aplicación de forma adecuada a nuestros intereses.
- 2. Una vez identificados, se nos mostrará la visión adecuada del programa, permitiéndonos trabajar según nuestros intereses. Tras la finalización de

nuestras tareas, nos desconectaremos de la aplicación y volveremos a la visión del grupo de usuario invitado.

4.2.- Diseño de la aplicación

En este apartado se llevará a cabo la descripción de cómo se ha desarrollado la aplicación: la base de datos que almacena toda la información, los distintos menús de los distintos grupos de usuarios, y los diversos formularios y estructuras que componen cada uno de los menús.

4.2.1.- Base de datos

Vamos a describir detalladamente la base de datos de nuestra aplicación, describiendo cada una de las tablas y demás elementos que la componen y las relaciones que entre estas existen. Al final del apartado se observa el diagrama entidad-relación (ERD) de la misma.

Nuestra base de datos consta de 23 tablas, aunque en el ERD se observan 25 debido a que aparecen dos tablas de la meta-base de datos (_user y _group) que están relacionadas con la tabla Person. A continuación detallaremos cada una de las tablas:

• **Person:** esta tabla almacena información personal acerca de los usuarios participantes. Solo guardamos información de este tipo de usuarios porque son los relevantes.

Esta tabla esta relacionada con la tabla _user de la meta-base de datos para relacionar los datos de cada usuario participante con usuario propiamente dicho. Se puede apreciar en el ERD que la relación es de tipo 1:0, ya que toda participante deberá tener su usuario pero no todo usuario será participante. Para esta relación se define el campo 'uid'. Además esta tabla se relaciona con las tablas Telephone, Email, Addres, Remarks, Message, todo y REL_paropro, tablas que iremos describiendo a continuación.

- **Telephone:** esta tabla permitirá almacenar los teléfonos de los participantes, tanto el número como una breve descripción que indique su naturaleza (casa, oficina...).
- **Email:** esta tabla permitirá almacenar las direcciones de correo electrónico de los participantes, asi como una breve descripción que indique su naturaleza (personal, trabajo...).
- Addres: esta tabla permitirá almacenar las direcciones de los participantes, asi como una breve descripción que indique su naturaleza (casa, oficina...).
- **Remarks:** esta tabla recogerá posibles notas o aclaraciones acerca de los usuarios participantes.
- Message: tabla que se usa para registrar los mensajes del servicio de mensajeria.

- todo: tabla en la que se almacenan las distintas notas que un usuario guarda.
- **Type:** esta tabla almacenará los tipos de proyectos que gestiona la aplicación, guardando el nombre del tipo y una breve descripción de este.
- **Project:** esta tabla almacenará los distintos proyectos que se gestionan mediante la aplicación, registrando de los mismos su nombre, el tipo al que pertenece (este campo establece una relación entre esta tabla y la tabla Type) y una breve descripción.
- **Estado:** esta tabla recoge los posibles estados en los que se pueden encontrar las tareas de métrica v3 en su realización en un determinado proyecto.
- **Proccess:** esta tabla recogerá los procesos en los que se divide la gestión de procesos según métrica v3. De cada uno de ellos guardará su nombre, así como un texto que explica en que consiste dicho proceso. Se relaciona con la tabla Activity como veremos a continuación.
- Activity: en esta tabla se almacenarán las diversas actividades en que se divide la gestión de proyectos según métrica v3. De cada actividad guardará su nombre, el proceso al que pertenece (mediante este campo se relaciona con la tabla Proccess) y un texto que explicará en que consiste dicha actividad. Esta tabla se relaciona con la tabla Task.
- Task: en ella se guardan las distintas tareas en que se divide la gestión de proyectos según métrica v3. De cada tarea almacenará su nombre, la actividad a la que pertenece (mediante este campo se relacionan esta tabla y la tabla activity) y un texto en el que se detallará en que consiste esta tarea.

Las relaciones de esta tabla con otras tablas, a parte de la ya comentada con la tabla Activity, dan lugar a nuevas tablas por ser relaciones M:N, por lo que serán comentadas al comentar dichas tablas.

- **Profile:** esta tabla recoge los grupos en los que se dividen los roles que participan en las tareas antes mencionadas, según métrica v3. Se relaciona con la tabla Rol como ahora indicaremos.
- **Rol:** en esta tabla se almacenan los diversos roles que participan en los proyectos que se gestionan según métrica v3. De cada rol se recoge el nombre del mimos, el grupo al que pertenece (mediante este campo se relacionan las tablas Profile y Rol) y una descripción del mismo.

Esta tabla tiene otras relaciones, pero también dan lugar a otras tablas, por lo que se comentarán al describir dichas tablas.

• **Tecnic:** esta tabla guarda las distintas técnicas usadas en las tareas en que métrica v3 divide la gestión de los proyectos. Para cada técnica guarda su nombre, una descripción de esta y un ejemplo.

Las relaciones de esta tabla dan lugar a otras tablas por lo que también serán comentadas posteriormente.

• **InProduct:** en ella se almacena los productos de entrada que se esperan en las tareas en que se divide un proyecto según métrica v3. De cada producto registra su nombre, una descripción del mismo y un ejemplo.

Las relaciones de esta tabla dan lugar a otras tablas por lo que también serán comentadas posteriormente.

• OutProduct: en ella se almacena los productos de salida que se producirán en las tareas en que se divide un proyecto según métrica v3. De cada producto registra su nombre, una descripción del mismo y un ejemplo.

Las relaciones de esta tabla dan lugar a otras tablas por lo que también serán comentadas posteriormente.

- **REL_taip:** en esta tabla se recogen las relaciones entre la tabla Task y la tabla InProduct, de forma que en cada tupla de la misma se indica un producto de entrada y una tarea. Así, esta tabla recoge para cada tarea que productos de entrada se esperan.
- **REL_taop:** en esta tabla se recogen las relaciones entre la tabla Task y la tabla OutProduct, de forma que en cada tupla de la misma se indica un producto de salida y una tarea. Así, esta tabla recoge que productos de salida se han de generar en cada tarea.
- **REL_rota:** esta tabla guarda las relaciones entre la tabla Task y la tabla Rol, de forma que en cada tupla recoge un rol y una tarea. Así en esta tabla se indica que roles participan en cada tarea.
- **REL_tate:** esta tabla almacena las relaciones entre la tabla Task y la tabla Tecnic, de tal manera que cada tupla de dicha tabla recoge una tarea y una técnica. De este modo, en esta tabla se nos indica que técnicas se usan en cada tarea.
- **REL_tyta:** en esta tabla se guardan las relaciones entre las tablas Task y Type, de forma que en cada tupla de la misma se indican un tipo de proyecto y una tarea. Así, en esta tabla se indica que tareas se llevan a cabo en cada tipo de proyecto.
- **REL_paropro:** esta tabla relaciona a las tablas Person, Project y Rol. Para cada tupla se guarda un participante, un proyecto y un rol, de forma que esta tabla recoge para cada proyecto, que participante intervienen en el y con que rol lo hacen.
- **REL_protaro:** esta tabla recoge la relación entre las tablas Project, Task y Rol. Por cada tupla se indica un proyecto, una tarea, un rol, un estado, una fecha y un archivo. De esta forma, en esta tabla se recoge por cada proyecto que tareas lo componen y que rol ejecuta cada tareas. Además guarda el estado en que se

encuentra la tarea, la última vez que se modifico y el resultado de la misma si tiene y el usuario que lo creo lo incluye.

Esta tabla es muy importante en la aplicación ya que es la que recoge la evolución de los proyectos según la van marcando los participantes en los mismos.

4.2.1.1.- Vistas

Una vista es una representación del contenido de las tablas de la base de datos que agiliza el acceso a los datos haciendo que el motor de WAINE tenga que trabajar menos y que la aplicación vaya mucho más rápida.

En el desarrollo de nuestra aplicación, para su uso mientras se ejecuta la misma, se crean las siguientes vistas:

• VIEW_prorolusu: esta vista guarda para cada proyecto que usuarios participan y con que rol. Es parecida a la tabla REL_paropro pero en este caso lo que se relaciona no es el participante, si no el usuario en si. Esto agiliza las consultas debido a que la persona que entra en la aplicación lo hace como usuario, y cuando se quieren filtrar consultas se hace por su usuario.

```
CREATE VIEW VIEW_prorolusu AS

SELECT REL_paropro.fkpro AS fkpro,

REL_paropro.fkrol AS fkrol,

Person.uid AS uid

FROM

REL_paropro, Person

WHERE

REL_paropro.fkpar=Person.pk;
```

• **VIEW_proacttas:** esta vista guarda en cada una de sus filas una tarea, la actividad a la que pertenece y el proceso al que pertenece. Con esto se consigue agilizar también la búsqueda, ya que no hay que acceder a tres tablas si no a una sola vista.

```
CREATE VIEW VIEW_proacttas AS

SELECT task.pk AS pk,

proccess.name as pname,

activity.name as aname,

task.name AS tname

FROM

proccess, activity, task

WHERE

activity.fkpro=proccess.pk

AND

task.fkact=activity.pk;
```

• VIEW_prorolper: esta vista relaciona los datos de interés de cada participante con los proyectos en los que este participa y el rol que el que lo hace. De esta forma, cuando queremos mostrar datos de las participantes que trabajan en un proyecto no hay que acceder a varias tablas.

```
CREATE VIEW VIEW_prorolper AS
  SELECT
     rel_protaro.fkpro as fkpro, Project.name as project,
     Rol.name as rol, task.name as task, estado.estado as
     estado, rel_protaro.p_out as p_out, rel_protaro.fecha
     as fecha, rel_protaro.f_fin as f_fin,
     rel_protaro.notas as notas, person.uid AS uid
  FROM
     REL_protaro left outer join rel_paropro on
     (rel_protaro.fkpro=rel_paropro.fkpro and
     rel_protaro.fkrol=rel_paropro.fkrol) left outer join
     Person on rel_paropro.fkpar=person.pk, Project, Rol,
     Task, estado
  WHERE
     rel_protaro.fkpro=project.pk AND
     rel_protaro.fkrol=rol.pk AND
     rel_protaro.fktask=task.pk AND
     rel_protaro.estado=estado.pk;
```

• **VIEW_tastec:** este caso es parecido al de la vista VIEW_proactas pero además se añaden a la relación las técnicas que se usan en cada tarea.

```
CREATE VIEW VIEW tastec AS
   SELECT
      Task.pk as pk,
     Task.name as tname,
     proccess.name as pname,
     activity.name as aname,
     Tecnic.pk AS fktec,
     Tecnic.name AS tecname,
     Tecnic.example AS example
   FROM
       proccess, activity, Task, Tecnic
   WHERE
       activity.fkpro=proccess.pk
      AND
       task.fkact=activity.pk
      AND
       Task.pk=REL_tate.fktask
       Tecnic.pk=REL_tate.fktec;
```

• **VIEW_tasrol:** este caso es parecido al de la vista VIEW_proactas pero además se añaden a la relación los roles que participan en cada tarea.

```
CREATE VIEW VIEW tasrol AS
  SELECT
     Task.pk as pk,
     Task.name as tname,
     proccess.name as pname,
     activity.name as aname,
     Rol.pk AS fkrol,
     Rol.name AS rname
   FROM
       proccess, activity, Task, Rol
   WHERE
       activity.fkpro=proccess.pk
       task.fkact=activity.pk
      AND
       Task.pk=REL_rota.fktask
       Rol.pk=REL_rota.fkrol;
```

• **VIEW_tasip:** este caso es parecido al de la vista VIEW_proactas pero además se añaden a la relación los productos de entrada que se esperan en cada tarea.

```
CREATE VIEW VIEW_tasip AS
   SELECT
     task.pk AS pk,
           task.name AS tname,
     proccess.name as pname,
     activity.name as aname,
     InProduct.pk as fkip,
     InProduct.name AS ipname,
     InProduct.example AS example
      proccess, activity, task, InProduct
   WHERE
        activity.fkpro=proccess.pk
      AND
       task.fkact=activity.pk
      AND
        Task.pk=REL_tain.fktask
       InProduct.pk=REL_tain.fkip;
```

- **VIEW_tasop:** este caso es parecido al de la vista VIEW_proactas pero además se añaden a la relación los productos de salida que se crean en cada tarea. (El código de esta vista es como el de la anterior pero en este caso se accedería a la tabla OutProduct).
- **VIEW_tytaro:** esta vista relaciona un tipo de proyecto con las tareas que lo componen y los roles con que se participa en dicha tarea.

```
CREATE VIEW VIEW_tytaro AS

SELECT REL_tyta.fktyp AS fkpro,

REL_tyta.fktask AS fktask,

REL_rota.fkrol AS fkrol

FROM

REL_tyta, REL_rota

WHERE

REL_tyta.fktask=REL_rota.fktask;
```

• VIEW_tec, VIEW_ip y VIEW_op: estas vistan se crean para agilizar el acceso a la información de las tablas Tecnic, InProduct y OutProduct.

4.2.1.2.- Trigger y Rule

Un trigger, o disparo, es un elemento SQL que da lugar a la ejecución de una serie de instrucciones cuando se cumpla una determinada condición. Por su parte, un rule es un elemento SQL que agrupa un conjunto de instrucciones SQL bajo un mismo nombre. De esta forma, en nuestra base de datos tendremos varios trigger y varios rule, uno por trigger, para controlar la relación entre las tablas Project, Task y Rol, la tabla REL_protaro.

La idea es que un tipo de proyecto estará compuesto por un conjunto de tareas, el cual se definirá en la creación del tipo. Cada proyecto será de un tipo, y tendrá las mismas tareas que ese tipo. Para evitar tener que rellenar dos veces lo mismo, evitando incluso riesgo de equivocaciones, se crean una serie de disparos y reglas que actuarán cuando operemos sobre la tabla Project, como veremos a continuación.

• **tri_protaro:** este disparo se ejecuta cuando se realiza un Insert en la tabla Project, es decir, cuando introducimos en dicha tabla un nuevo proyecto. Su función es añadir en la tabla REL_protaro las relaciones entre proyecto, tarea y rol en función del tipo de proyecto del que se trate.

Este trigger llamará al rule *func_protaro_ins* el cual se encargará de ejecutar las instrucciones necesarias para conseguir lo indicado, de forma que al final del mismo, la tabla REL_protaro contendrá, además de lo que tuviese, para el nuevo proyecto que tareas se ejecutan en el y con que rol/es se participa en cada tarea.

```
CREATE TRIGGER tri_protaro AFTER INSERT
ON project FOR EACH ROW
EXECUTE PROCEDURE func_protaro_ins ();

CREATE OR REPLACE FUNCTION func_protaro_ins () RETURNS trigger AS'
BEGIN

INSERT INTO REL_protaro (fkpro, fktask, fkrol)
SELECT NEW.pk as fkpro, fktask, fkrol FROM
VIEW_tytaro WHERE fkpro=NEW.fktype;
RETURN NEW;
END;
'LANGUAGE 'plpgsql';
```

• **tri_protaro_del:** este trigger se dispara cuando ejecutamos la instrucción Delete sobre la tabla Project. El objetivo de este trigger es borrar las referencias, en la tabla REL_protaro, a un proyecto cuando este sea eliminado se nuestra aplicación.

Este trigger llama al rule *func_protaro_DEL* que se encarga de ejecutar las instrucciones adecuadas para que se cumpla lo antes mencionado.

```
CREATE TRIGGER tri_protaro_del BEFORE DELETE
ON project FOR EACH ROW
EXECUTE PROCEDURE func_protaro_DEL ();

CREATE OR REPLACE FUNCTION func_protaro_DEL () RETURNS trigger AS'
BEGIN

DELETE FROM REL_protaro WHERE fkpro = OLD.pk;
RETURN OLD;

END;
'LANGUAGE 'plpgsql';
```

• **tri_protaro_upd:** la función de este trigger, que se ejecuta cuando realizamos un Update sobre la tabla Project, es cambiar las relaciones registradas en la tabla REL_protaro cuando sea necesario, es decir, cuando cambie el tipo de un proyecto.

Para llevar a cabo su tarea este trigger llama al rule *func_protaro_upd*, el cual borrará todos los registros referidos al tipo antiguo de la tabla REL_paropro e insertará las relaciones que se deben cumplir debido al nuevo tipo del proyecto.

Aunque no es muy normal que un proyecto cambie de tipo, se incluye este trigger por si se diera el caso.

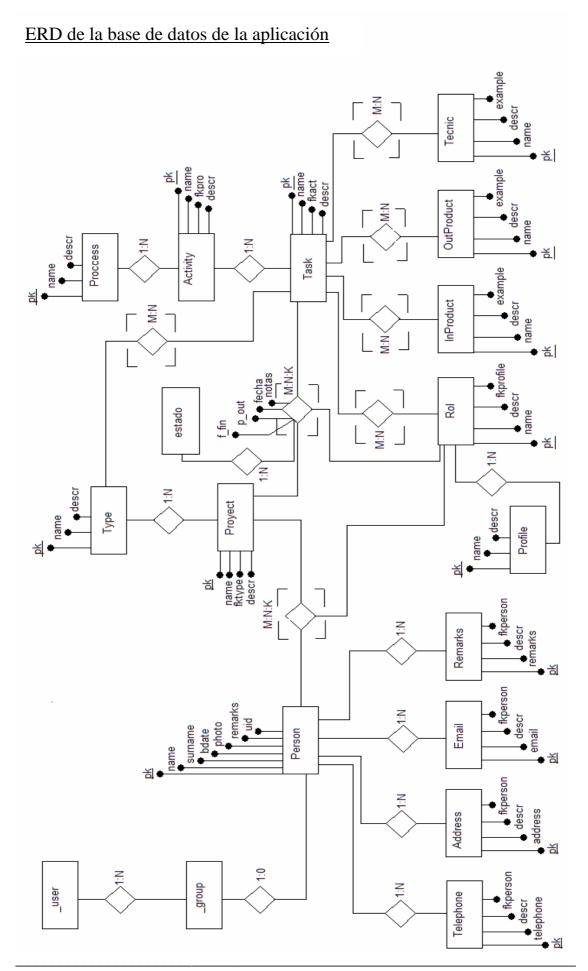
```
CREATE TRIGGER tri_protaro_upd AFTER UPDATE
ON project FOR EACH ROW
EXECUTE PROCEDURE func_protaro_upd ();

CREATE OR REPLACE FUNCTION func_protaro_upd () RETURNS trigger AS'
BEGIN

DELETE FROM REL_protaro WHERE fkpro = OLD.pk;

INSERT INTO REL_protaro (fkpro, fktask, fkrol)
SELECT NEW.pk as fkpro, fktask, fkrol FROM
VIEW_tytaro WHERE fkpro=NEW.fktype;
RETURN NEW;

END;
'LANGUAGE 'plpgsql';
```



4.2.2.- Diseño gráfico

Previo al desarrollo de la aplicación, una vez definida la base de datos, hemos de decidir que queremos mostrar de la misma en nuestra aplicación y como lo queremos mostrar, es decir, hemos de definir gráficamente nuestra aplicación.

En nuestro caso, una vez definido el número de tipos de participantes que accederían a la aplicación y determinado con ellos el número de menús de la misma, se pasa a diseñar uno por unos dichos menús.

En cada opción hemos de definir a que datos de que tablas de la base de datos queremos acceder, y como los queremos mostar: un formulario para rellenar, una tabla, un combo, .etc. En la siguiente imagen podemos ver un ejemplo de cómo se lleva a cabo esta tarea.

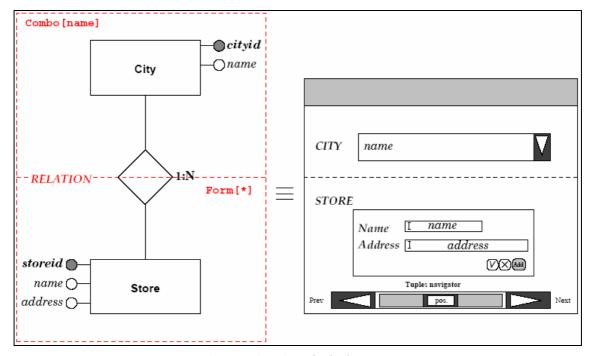


Figura 10.- Ejemplo de diseño

En este ejemplo podemos ver como se representa la relación entre las tablas City y Store. En la parte superior mostramos en un combo la información referente a la tabla City, en este caso el nombre, mientras que en la parte inferior mostraremos la información referente a la tabla Store a través de un formulario.

Según el valor de City que escojamos en el combo, se nos mostrarán abajo los almacenes relacionados con esa ciudad. Aunque en el ejemplo no se ve, debería aparecer también un navegador en la parte inferior en el que se muestren los almacenes relacionados a los que podemos acceder.

Así, para cada opción de los diversos menús, al igual que en este ejemplo decidimos que queremos mostrar y como, y se diseña gráficamente para su posterior codificación en ASL, teniendo en cuenta las relaciones que se producen en la base de datos.

4.2.3.- Interfaces de usuarios

Tras la descripción de nuestra base de datos, a continuación vamos a pasar a describir las distintas interfaces que se muestran a los distintos usuarios que componen nuestra aplicación.

4.2.3.1.- Menú de invitado

Este menú es el más básico de los tres de los que dispone la aplicación. En este caso se muestran 3 submenús, cada uno de ellos con una serie de posibles opciones, como podemos observar en la siguiente imagen.



Figura 11.- menú de invitado

Como podemos ver en la figura los tres submenús son: Formación, Misc e Identificarse.

El submenú Misc está presente en todos los menús, por lo que solo lo comentaremos aquí. Este submenú nos presenta dos opciones, dos enlaces. El primero de ellos, Autor, nos permite ver información a cerca del autor y la aplicación. El segundo de ellos, Desconectarse, nos permite salir de la aplicación y no se muestra en el menú de invitado ya que a el se accede sin conectarse.

El submenú Conectarse nos permite identificarnos y acceder a otra visión de la aplicación, participante o root.

Info	
Name	Value
appname	m3m
ver	0.1
abstract	Una aplicaciOn para gestionar proyectos en Metrica v3
author	A.L.Delgado y J.A.Jimenez
email	aldelgado@gmail.com, elautenticopanzeco@hotmail.com
date	5-01-2007

Figura 12.- Página Autor

Por su parte, el submenú Formación nos da una serie de opciones que nos llevarán a formularios que nos permitirán obtener información sobre métrica v3, facilitando el aprendizaje de la misma.

• Los enlaces "Ver un" nos permitirán acceder a la información de un proceso, una actividad o una tarea concreta, mientras que los enlaces "Lista de" nos permitirán acceder a la información de todos los procesos, actividades o tareas.

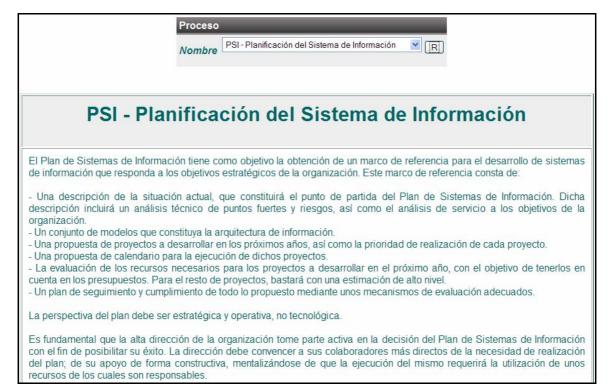


Figura 13.- Ejemplos de "Ver un proceso"

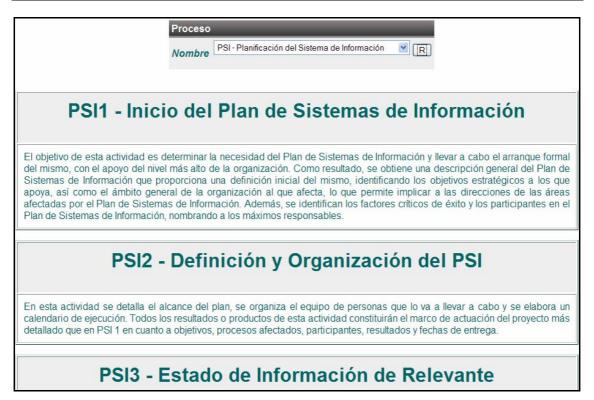


Figura 14.- Ejemplos de "Lista de actividades

- "Técnicas, roles y productos de una tarea" nos permitirá ver las relaciones de una tarea con los elementos mencionados en el título del enlace.
- "Lista de roles" nos mostrará la información de los diversos roles.
- Las opciones "Tareas por" nos permitirán ver en que tareas se encuentran presentes un rol, una técnica, un producto de entrada o un producto de salida.

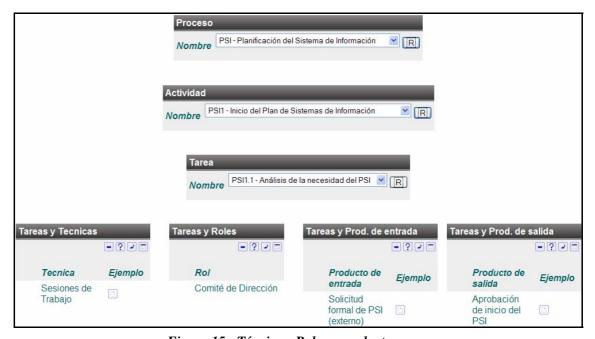


Figura 15.- Técnicas, Roles y productos



Figura 16.- Ejemplos de Tareas por Técnica y Lista de roles

4.2.3.2.- Menú de root

Este menú permite la gestión de la aplicación. Está subdividido en seis submenús, como podemos ver en la siguiente imagen, que pasaremos a comentar a continuación.



Figura 17.- Menú de root

El submenú m3m, mediante las opciones que nos proporciona, nos permite la gestión de toda la información relacionada con métrica v3:

• La opción "Procesos" nos permitirá gestionar los procesos en que conforman la gestión de un proyecto según métrica v3.

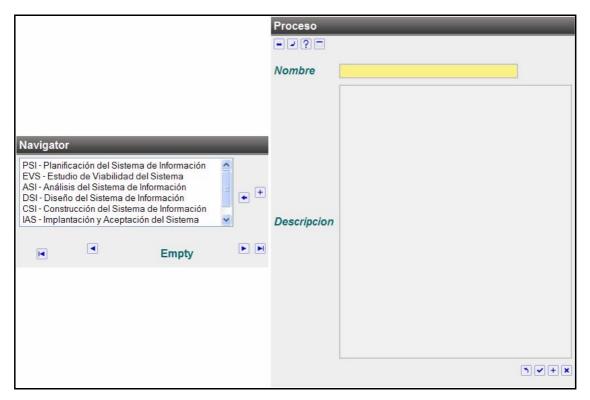


Figura 18.- Procesos de métrica v3

- La opción "Actividades" nos permitirá gestionar que actividades conforman la
 gestión de un proyecto según métrica v3, indicando además el proceso al que
 pertenece cada actividad. Para ello primero se seleccionará un proceso mediante
 un combo y luego mediante un formulario podremos añadir que actividades
 forman parte de ese proceso.
- La opción "Tareas" proporciona la capacidad de definir las tareas en que se subdivide la gestión de un proyecto según métrica v3, indicando además el proceso y la actividad a la que pertenecen cada tarea. Para ello, de forma similar a como se hacía en la opción "Actividades" se selecciona mediante un combo el proceso, tras lo cual se nos muestra en otro combo las actividades que pertenecen a dicho proceso. Seleccionamos la actividad que queramos y en el formulario inferior se añaden las tareas que compondrán dicha actividad. Debido a la similitud con la opción anterior mostraremos solo la imagen de esta que es algo más compleja.

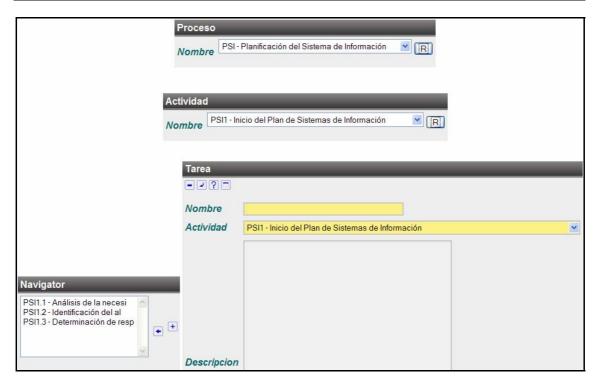


Figura 19.- Pantalla Tareas

 Mediante la opción "Rol" se nos permite definir que grupos de roles existen, además de poder definir los roles y a que grupo pertenecen. Esto se realiza mediante dos formularios, como podemos observar en la siguiente figura. El formulario de roles se filtra según el valor seleccionado en el formulario profile.

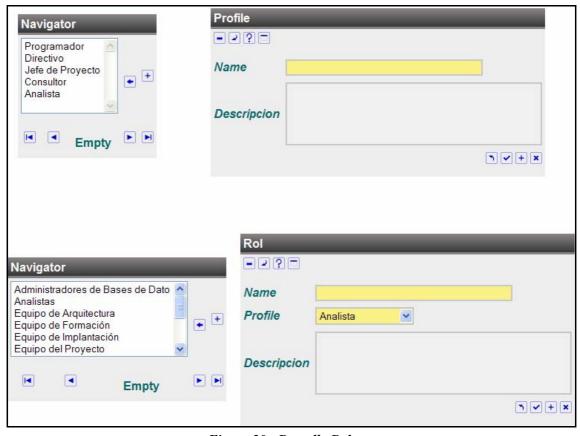


Figura 20.- Pantalla Roles

• La opción "Tecnicas" nos permite definir las técnicas que se usan en las distintas tareas en que se divide la gestión de un proyecto según métrica v3.

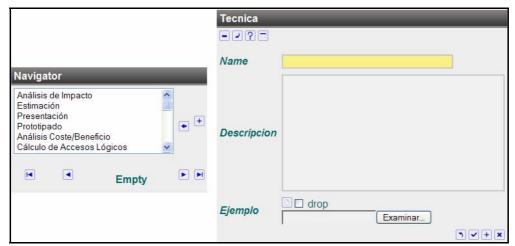


Figura 21.- Pantalla Tecnicas

- Las opciones "Productos de entrada" y "Productos de Salida" permiten registrar los ambos tipos de productos. Sus pantallas son muy similares a la pantalla Técnicas, por lo que no se incluye imagen.
- Mediante la opción "Relaciones" se nos permite indicar las relaciones de las tareas con los distintos elementos antes mencionados: técnicas, roles, productos de entrad y productos de salida. Mediante tres combos se indicará el proceso y la actividad a los que pertenece la tarea, y la tarea propiamente dicha. Una vez indicado, mediante las cuatro pestañas inferiores podremos añadir técnicas que se usan en dicha tarea, roles que participan, productos de entrada que se esperan para esa tarea y productos de salida que se deben generar al final de la misma. En la siguiente imagen podemos observar esta pantalla.



Figura 22.- Pantalla Relaciones

El submenú Proyectos nos permite administrar los aspectos de la aplicación relacionados con los proyectos en si:

• Mediante la opción "Crear Tipos" se nos da la posibilidad de crear los tipos de los que pueden formar parte los proyectos. Para ello en la parte superior se nos muestra un formulario, en el que indicaremos nombre y descripción del tipo, y en la parte inferior se nos muestra una tabla, dependiente del tipo que tengamos seleccionado, y que nos muestra las tareas que conforman dicho tipo, permitiéndonos añadir, quitar o modificar las tareas que componen cada tipo.



Figura 23.- Pantalla Crear Tipos

- La opción "Creación de proyectos" nos permite crear los proyectos que se gestionarán mediante la aplicación. Primeramente se filtrará el tipo de proyecto mediante un combo y luego se rellena un formulario en la parte inferior, de forma similar a como se hacia en "Actividades" o en "Tipos" por lo que no mostraremos esta pantalla.
- A través de la opción "Editar proyectos" se nos permite añadir o eliminar tareas de un proyecto de forma que no habrá que crear tipos concretos para cada proyecto. Para editar el proyecto se selecciona un proyecto mediante un combo, tras lo cual se mostrará en la parte inferior, mediante una tabla con dos combos, las tareas que componen el proyecto y con el rol que se llevan a cabo, permitiendo eliminar o añadir pares de este tipo.

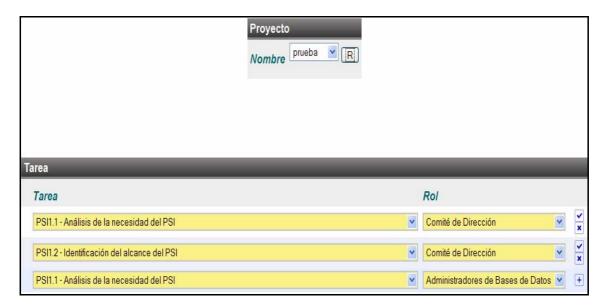


Figura 24.- Pantalla Editar proyectos

 Mediante la opción "Asignación de funciones" se nos da la posibilidad de tras seleccionar un proyecto mediante un combo, en la parte inferior, mediante una tabla indicar quien participa en ese proyecto y con que rol lo hace, como podemos ver en la siguiente figura.



Figura 25.- Pantalla Asignación de Funciones

• Mediante la opción "Definir estados de las tareas" se permite al administrador definir los posibles estados en los que se puede encontrar una tarea.

Los submenús "Cosas por hacer" y "Mensajes" son iguales a los del menú de participante y se definirán en el siguiente apartado, referente a dicho menú.

El submenú Administración de usuarios nos permite gestionar los usuarios que tomarán parte de nuestra aplicación:

• La opción "Usuario administrador" nos permite editar el nombre y la contraseña de dicho usuario.



Figura 26.- Pantalla Usuario Administrador

• Mediante la opción "Usuarios Participantes" se nos permite crear y editar usuarios participantes además de añadir los datos personales de estos, que son los que nos interesan. Para ello en la parte superior tenemos un formulario que nos permite crear y editar al usuario. Según el usuario que tengamos escogidos se filtrarán en el formulario inferior los datos asociados a ese usuario, y cuando seleccionemos que datos queremos se filtraran en los tabs inferiores otros datos de interés del participante. En la siguiente figura podemos observar todo esto.

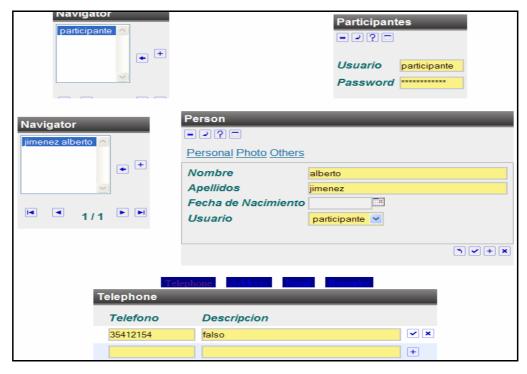


Figura 27.- Pantalla Usuarios Participantes

Por último, el submenú Misc que ya fue comentado en el apartado anterior referido al menú de invitado.

4.2.3.3.- Menú de participante

Este menú permite a los usuarios participantes acceder a la información sobre los proyectos en los que ellos participan, les permite introducir información sobre las tareas que realizan en los mismos y además les permite obtener información sobre métrica v3 en caso de necesitarla.

El menú de participante se subdivide en 6 submenús, como se puede ver en la siguiente figura, y que pasaremos a describir a continuación.



Figura 28.- Menú de participante

Cuando entramos como usuario participante, lo primero que se nos muestra es un resumen de nuestras actividades. Este resumen está construido como un struct de tipo tab en el que se indican tres tabs: Mis tareas, Bandeja de entrada y notas. Según al que accedamos nos mostrará la información relativa al tab seleccionado.

El submenú consultas nos permite acceder a la información a cerca de los proyectos en los que estamos trabajando, mientras que el submenú proyectos nos permite introducir información sobre las tareas en las que participamos. Estos submenús nos da las siguientes opciones:

• La opción "Proyectos por tipo" nos muestra todos los proyectos del tipo elegido en los que participamos. Para ello en la parte superior se muestra un combo sobre el que elegiremos el tipo deseado, y una vez seleccionado este, en la parte inferior, mediante una lista, se nos mostrarán todos los proyectos es los que participamos de ese tipo. En la siguiente figura se nos muestra esta pantalla.



Figura 29.- Pantalla Proyectos por tipo

- Mediante la opción "Funciones por proyecto" podemos ver que tareas tenemos que realizar en un determinado proyecto y con que rol participamos en ese proyecto y en cada tarea. Para ello, en la parte superior mediante un combo seleccionamos la información de que proyecto, de entre los que participamos, queremos ver, y en la parte inferior se nos mostrará una lista con las tareas que hemos de llevar a cabo en ese proyecto y el rol con el que participamos en esas tareas. Debido a la similitud que presenta con la opción anterior no mostraremos esta pantalla.
- A través de la opción "Funciones por rol" podemos ver en que proyectos participo con ese rol y que tareas hemos de llevar a cabo en dicho proyecto por tener ese rol asignado. La forma de realizar esta pantalla es igual a la anterior pero en este caso en el combo mostramos y seleccionamos uno de los roles que ejecutamos en algún proyecto, y en la parte inferior, en la lista se nos muestra en que proyecto/s participamos con ese rol, y que tareas hemos de ejecutar en dicho proyecto por tener ese rol. Igualmente que en caso anterior tampoco pondremos figura de este caso por su similitud con los dos anteriores.
- La opción "Estado de mis tareas" nos permitirá ver y modificar el estado de las tareas que tenemos que llevar a cabo y añadir los productos de salida cuando estos se tengan. En esta pantalla se nos vuelve a mostrar un combo en la parte superior en el que deberemos elegir aquel proyecto sobre el que queramos trabajar. Una vez seleccionado este, en la parte inferior se nos mostrarán las tareas que ejecutamos en este proyecto y con que rol participamos en ella, pero además, junto a cada una de las tareas aparecen tres campos editables, el estado en que se encuentra la tarea, la fecha de la última modificación de la misma, y un blob-file que recoge el producto de salida de dicha tarea. En la siguiente figura se nos muestra un ejemplo de esta pantalla.



Figura 30.- Pantalla Estado de mis tareas

• La opción estado de proyectos nos permite ver información sobre las tareas de los proyectos en los que participamos. Primeramente, mediante un combo, seleccionamos el proyecto sobre el que queremos obtener la información, y una vez seleccionad, se nos muestra en la parte inferior mediante una tabla las tareas del proyecto junto con que roles la ejecutan, el estado en que se encuentra, la fecha de su última modificación y la localización del producto de salida si estuviese. Esta pantalla se puede apreciar en la siguiente figura.



Figura 31.- Pantalla Estado de proyectos

• Mediante la opción "Información de los proyectos" se nos permite acceder a los datos más relevantes para nuestros intereses de los demás participantes de aquellos proyectos en los que yo trabaje. Igual que en el caso anterior, en este caso también seleccionaremos mediante un combo el proyecto del que queremos obtener la información, tras lo cual, en la parte inferior, se nos mostrará el nombre y apellidos de los participantes, su usuario, su foto, su teléfono, su

dirección de correo electrónico y toda la información relativa a estos que tengamos. La forma en la que se muestra la información es igual a la forma en que se introducen los datos en le menú de root por lo que no se muestra imagen.

El submenú "Cosas por hacer" presenta una única opción llamada notas y que nos permitirá crear notas acerca de las cosas que nos quedan pendientes en un proyecto, cosas que queremos recordar...

En esta pantalla se nos permite por un lado crear la nota, dar una descripción de lo que se quiere, y después mediante un botón se nos abre en una ventana un formulario referente a la nota que nos permite añadir otros datos tales como las fechas de creación y finalización, adjuntar un archivo...

En la siguiente figura podemos observar las dos partes mencionadas de esta opción.

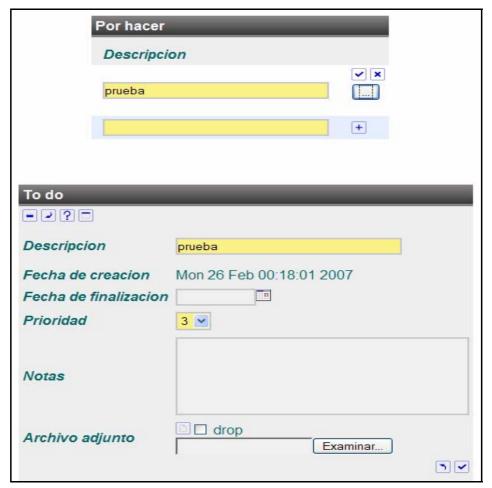


Figura 32.- Pantallas de Notas

El submenú Mensajes presenta tres opciones que nos van a permitir dar servicio de mensajería entre usuarios. Por un lado una opción nos mostrará la bandeja de entrada, otra opción nos permitirá crear mensajes y la última opción nos permitirá enviar dichos mensajes a otros usuarios.

Finalmente, los submenús Formación y Misc ya fueron comentados en el apartado sobre el menú de invitados.

4.3.- Implementación del código

En ese apartado pasaremos a comentar la estructura del código de la aplicación así como algunos ejemplos de los elementos usados en dicho código para crear la estructura que queremos.

El código completo de la aplicación lo encontraremos al final de esta memoria, pero con este apartado se busca facilitar un poco más la comprensión del mismo.

La estructura del archivo sería la siguiente:

- Por tratarse de un archivo XML, las dos primeras líneas serán las típicas de todo documento XML y las cuales podemos observar a continuación.

```
<?xml version='1.0' ?>
<!DOCTYPE asl PUBLIC "-//ITI//DTD ASL 0.6 //EN"
   "/usr/local/lib/waine-0.1.2007-02-01/lib/asl.dtd" >
```

Mediante estas dos etiquetas se indica la versión de XML y la localización del DTD que seguirá nuestro documento.

- A continuación, entre las etiquetas <asl> y </asl> irá todo el código de nuestra aplicación.
- Lo primero que encontramos tras la etiqueta <asl> son los elementos include. Estos elementos nos permitirán el uso de las librerías que incluyamos con ellos, pudiendo usar sus módulos en nuestro documento. En nuestra aplicación usamos tres elementos include para incluir los módulos message, todo y meta, como podemos observar en el código que se muestra a continuación.

- A continuación, entre las etiquetas <head> </head> se definen los elementos meta que serán usados por la librería meta.asl

 A continuación se definen los distintos grupos de usuarios que formarán parte de nuestra aplicación, así como los usuarios que inicialmente formarán parte de cada grupo, como podemos observar a continuación.

- A continuación, se definen los distintos main, tantos como sean necesarios. En su interior, cada main contendrá sus correspondientes elementos menu y a estos a su vez contendrán sus elementos option si fuesen necesarios, tal y como vimos en el apartado 2.5.1.2.- Menu.

```
<main id="main1">
       <menu caption="Proyectos" img="usr.jpg">
           <option caption="Project" call="st_project"/>
           <option caption="Project Config"</pre>
                             call="st_project"/>
       </menu>
      <menu caption="Proceso">
           <option caption="Process" type="form">
               <param name="formid" value="frm_process"/>
               <param name="navigator_position" value="W"/>
               <param name="navigator fields" value="1"/>
               <param name="navigator width" value="3"/>
               <param name="button data" value="1"/>
           </option>
           <option caption="Activity" type="relation">
               <param name="form_split" value="rows=100,*"/>
               <param name="formid" value="frm_process"/>
            </option>
      </menu>
</main>
```

- Finalmente, tras la definición de los elementos menú correspondientes para la creación de los distintos menús de los usuarios se crean los formularios y las estructuras, elementos form y struct, necesarios, para terminar cerrando la etiqueta asl.

Observando el código de ejemplo del menú, podemos ver como en los option en los que se llama a una estructura se hace uso del elemento call y no se usan elementos option, mientras que en los que se llama al formulario directamente si lleva elementos option que se usan para definir ciertas características del formulario.

4.4.- Configuración

En este apartado vamos a comentar los archivos que se usan para configurar la aplicación de forma que tenga el aspecto que nosotros queremos y trabaje con el tipo de datos que queramos. Estos archivos son los siguientes:

4.4.1.- db.cfg

Este archivo guarda la configuración de la base de datos. En el indicamos el driver que se usará, en nuestro caso 'dspg.inc' por tratarse de una base de datos desarrollada en postgresql. A continuación se indican el nombre de la base de datos, el servidor donde se encuentra, el usuario y el password mediante el cual se realizará la conexión a la base de datos.

```
<?php

$DBDRIVER="dspg.inc";

$DBNAME="m3m";

$DBSERVER="localhost";

$DBPORT="";

$DBUSER="alberto";

$DBPASSWD="";

$DBPASSWD="";

$DBOPTIONS="";

?>
```

4.4.2.- mdb.cfg

En este archivo indicaremos el tipo de base de datos que se usará para la metabase de datos. En nuestro caso, como la base es sqlite, se indicará que se usará el driver dssglite.inc. Además se indica la ruta al archivo que contendrá la meta-base de datos.

```
<?php

$DBDRIVER="dssqlite.inc";
$DBFILE="./MDB";

?>
```

4.4.3.- menu.cfg

Mediante este archivo podemos editar las características que tendrá el menú de nuestra aplicación. Mediante este archivo podemos adaptar multitud de menús para nuestra aplicación. El contenido de este archivo se mostrará en el apartado siguiente.

4.4.4.- themes

A través de estos archivos podremos configurar la apariencia de determinados struct, añadiendo un elemento param en estos que referencia al theme. En nuestro caso se crean 3 archivos de este tipo para adecuar la forma de mostrar la información de Métrica v3.

Al igual que en caso anterior se mostrará el contenido de estos tres archivos en el apartado siguiente referente al código de la aplicación.

4.5.- Estimaciones temporales

En este apartado se comentarán las estimaciones sobre los tiempos en que se han llevado a cabo las diversas tareas a través de las cuales hemos llegado al desarrollo de nuestro proyecto.

4.5.1.- Aprendizaje de ASL

Como se ha comentado en apartados anteriores ASL es el lenguale mediante el cual se desarrollan las aplicaciones en WAINE. Éste es un lenguaje XML definido mediante su correspondiente DTD.

Evidentemente, el tiempo de aprendizaje de este lenguaje depende mucho de los conocimientos de XML que se tengan. En mi caso, que conocía el lenguaje pero de forma muy básica, el aprendizaje fue de aproximadamente una semana (en horas serían unas 20 horas). En esa semana realizaba ejemplos e intentaba ver que podía construir con este lenguaje. Claro esta que no aprendí el lenguaje en toda su profundidad, pero si lo fundamental para crear la aplicación, y después, a lo largo del desarrollo y depuración del proyecto he profundizado mucho más en la riqueza y capacidad del lenguaje.

4.5.2.- Desarrollo de la base de datos

El desarrollo de la base de datos es fundamental para crear una aplicación WAINE. Esta debe recoger toda la información que necesitemos y que vayamos a mostrar.

El desarrollo inicial de la base de datos me ocupo 2 días (en horas, unas 4 horas), pero igual que en caso anterior, la base de datos se ha ido modificando a lo largo del desarrollo de la aplicación, añadiéndole nuevos aspectos que nos fuesen haciendo falta, como por ejemplo las vistas.

4.5.3.- Funcionamiento de WAINE

El funcionamiento en si de la herramienta es sencillo. En primer lugar hemos de crear el directorio en el que residirá la aplicación mediante la herramienta de WAINE mkapp, para posteriormente crear la aplicación en si a partir de nuestro archivo asl mediante la herramienta de WAIN E asl2mdb.

Como resultado de ejecutar esta segunda herramienta obtendremos una serie de mensajes indicándonos los distintos pasos en la creación de la aplicación si todo es correcto, o un texto en el que se nos indican los fallos que se encuentran en el archivo asl y en que parte del mismo se encuentran, lo que facilita mucho la corrección de archivos asl.

Dicho esto no creo que se pueda considerar importante en tiempo de aprendizaje para usar la herramienta en si misma.

4.5.4.- Diseño y construcción de la aplicación

Esta parte es la más costosa de la aplicación, y me ha llevado unos dos meses (en horas, unas 100 horas). Hay que tener en cuenta que para un novato en WAINE, como es mi caso, esta parte no consiste solo en desarrollar la aplicación, que ya es costoso, si no que en ellas profundizas en ASL, modificas la base de datos, etc.

Evidentemente, una persona que ya conozca la herramienta y que le dedique el tiempo seguido a realizar la aplicación disminuiría bastante los tiempos de realización de la misma, por lo que considero que puede ser WAINE puede ser una herramienta muy interesante para el desarrollo de aplicaciones Web.