PROYECTO FIN DE CARRERA

GESTIÓN DE CINES DE ESPAÑA MEDIANTE UNA INTERFAZ VOCAL : "SEVENTHART"

Realizado por :
HASSNAE BEKKALI

Profesor tutor :
DAVID PICO VILA

Curso 2009/2010

Indice

Dedicatoria…….…...3

Agradecimientos…….…..4

Introducción…….…....5

PRIMERA PARTE : PRESENTACIÓN DEL TRABAJO

1. Objetivo del trabajo…….……...8

2. Organización de la memoria……..……...8

SEGUNDA PARTE : FASE CONCEPTUAL

3. F ase conceptual……...….11

3.1. Diagrama de clases…….…….…....11

3.2. Diagrama de flujo…….…....17

3.3. Diseño de la base de datos……33

3.3.1. Alojamiento de la base de datos……...33

***3.3.2.* Estructura de la base de datos…….33**

TERCERA PARTE : FASE DESARROLLO

4. Fase del desarrollo…….41

4.1. Desarrollo de la IVR……..45

4.1.1. Gramáticas……....……45

4.1.1.a. Gramáticas estáticas……45

4.1.1.b. Gramáticas DTMF……...46

4.1.1.c. Gramáticas dinámicas……47

4.1.1.d. Gramáticas propias del VXML o gramáticas builtin……..48

4.1.2. Tecnología utilizada…….....50

4.1.2.a. Vxml…….……...50

4.1.2.b. Javascript……...…57

4.1.2.c. Java y Jsp……...63

4.1.2.d. Json…….63

4.2. Desarrollo de la WEB……65

4.2.1. WEB : consultas……...65

***4.2.2.* WEB : insertar datos……...73**

CUARTA PARTE : FASE DE PRUEBAS

5. Fase de pruebas…….75

Test y traza de ejecución……..….....75

Conclusión……..…81

Bibliografía…….…..82

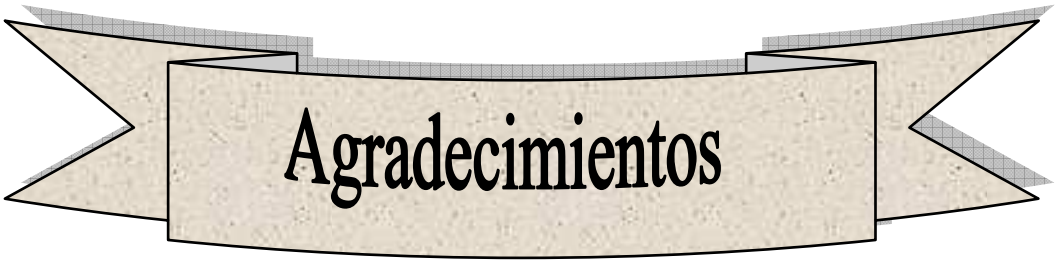
Glosario de términos……83

Dedicatoria

Dedico este trabajo a:

- *Mis padres porque todo lo que soy es gracias a dios y a ellos.*
- *Mi profesor David por su apoyo, su ayuda y su amabilidad.*
- *Mis hermanos por ser mis compañeros incondicionales.*
- *Mi futuro marido por su gran ayuda y sus consejos.*
- *Todos mis amigos y aquellos que me quieren.*

Hassnae



Agradecimientos

En primer lugar, expreso mi gratitud a la comisión docente de la Facultad de Informática por haber aceptado el tema desarrollado. De igual forma, agradezco a mi supervisor Dr. David Picó Vila por sus valiosas aportaciones y el tiempo dedicado al desarrollo de este trabajo.

También hago un agradecimiento especial a Youssef por sus consejos, su gran apoyo y por compartir su conocimiento. A todas las personas que facilitan información por internet, consejos y tutoriales.

Finalmente agradezco a todas las demás personas que han compartido de una u otra forma, el proceso de elaboración de este proyecto.

Introducción

En los últimos años ha surgido un nuevo tipo de interfaces hombre-máquina que combina varias tecnologías de habla, con el fin de permitir a las personas la interacción vocal con los ordenadores.

Hoy en día, muchas personas están familiarizadas con las aplicaciones vocales de respuesta automática (IVR), como son por ejemplo las aplicaciones de atención al cliente o banca telefónica. Estas aplicaciones hacen uso de recursos como el reconocimiento de voz y la síntesis de voz para obtener información del usuario y proporcionarle la información requerida, respectivamente.

VoiceXML es un lenguaje basado en XML desarrollado por la W3C para la creación de diálogos vocales que hacen uso de recursos de reconocimiento de voz, síntesis de voz, reconocimiento DTMF, audio digital y grabación de audio en el contexto de aplicaciones telefónicas o interacción a través de la voz en general. Otra característica muy interesante del VoiceXML es que proporciona los mecanismos para la creación de diálogos de iniciativa mixta, que hacen posible una forma de interacción con el usuario mucho más rica. La gran ventaja en el uso de VoiceXML es que se ha convertido en el estándar más utilizado para el diseño de aplicaciones vocales, proporcionando una arquitectura abierta para la programación de este tipo de aplicaciones frente a los antiguos sistemas IVR cerrados o propietarios.

Los sistemas iniciales estaban muy limitados en cuanto al tipo de frases que los usuarios podían pronunciar y al tipo de tareas a realizar; sin embargo, durante las

tres últimas décadas su complejidad ha aumentado notablemente, permitiendo una interacción más natural y cercana a la humana. No obstante, y a pesar de los logros alcanzados, su funcionalidad sigue estando limitada a dominios de aplicación claramente restringidos, que permiten incorporar conocimiento y expectativas acerca de las posibles palabras, tipos de frases e intenciones de los usuarios.

Una de las causas del éxito de VoiceXML es el hecho de haber llevado las ventajas del desarrollo WEB y el acceso a contenidos WEB al desarrollo de aplicaciones IVR. De esta forma desarrollar una aplicación Vocal con VoiceXML se convierte en algo muy similar y casi análogo a desarrollar un formulario WEB utilizando HTML. En más adelante, veremos con detalle las características de este lenguaje.

PRIMERA PARTE
PRESENTACIÓN DEL TRABAJO

1. *Objetivo del trabajo :*

Este trabajo tiene como finalidad:

Diseñar una interfaz vocal capaz de gestionar los cines de España, que lleva el nombre de “SeventhArt”.

Esta interfaz permite a los usuarios hacer reservas en los distintos cines del país mediante llamadas telefónicas y de forma sencilla.

Se trata de una interfaz dirigida a personas de todas las generaciones, a diferencia de las aplicaciones clásicas, con interfaces web, que implican tener mucha idea de navegar por internet, y dedicar mucho tiempo para acceder a los distintos servicios de las empresas.

2. *Organización de la memoria :*

La estructura del proyecto presentado en esta memoria es la siguiente:

En principio, se hace una introducción sobre las aplicaciones IVR y sobre el lenguaje VXML. También se dan a conocer los objetivos del trabajo desarrollado.

En el capítulo 2, se describe con detalle la parte analítica del sistema. Se presenta una descripción detallada del diagrama de clases y del flujo de llamada, donde se reflejan los distintos caminos que sigue el usuario a la hora de comunicarse con la interfaz.

Posteriormente, se realiza una descripción de la base de datos con sus distintas tablas y los distintos campos. Al mismo tiempo, se cuenta en qué servidor esta alojada la base de datos para posteriores accesos.

En el capítulo 3, se presenta la parte del desarrollo completo del sistema. Se incluyen las gramáticas usadas y sus distintos tipos. También se presenta la tecnología que fue utilizada para llevar a cabo la implementación de la interfaz.

Esta tecnología incluye tanto los lenguajes como las herramientas.

También se incluyen partes del código con aclaraciones, para explicar las características del lenguaje VXML y sus propiedades.

Finalmente, se detallan las conclusiones, aportaciones personales y un glosario de los distintos términos que se han usado en este trabajo y que resultan nuevos para el lector.

SEGUNDA PARTE
FASE CONCEPTUAL

3. Fase conceptual:

Resumen: En este capítulo, se proporciona un panorama general sobre los elementos del sistema desarrollado. El diseño del esquema conceptual se ha hecho mediante la técnica de diagramas de clases.
Asimismo, en la segunda sección del capítulo se explica el diagrama de flujo donde están reflejados todos los posibles caminos que se podrían dar durante una posible llamada al sistema.
Finalmente, la tercera sección, consiste en describir el proceso de creación de la base de datos a partir de nuestro esquema conceptual y también su alojamiento en un servidor para posteriores accesos. Se presentan todas las tablas de nuestra base de datos con todos los atributos y las restricciones.

3.1. Diagrama de clases:

El diagrama de clases es un tipo de diagrama que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crean el diseño conceptual de la información que se manejará en el sistema y los componentes que se encargan del funcionamiento y la relación entre uno y otro.
En la [Fig.1] se muestra el diagrama de clases de nuestro sistema. Podemos observar que existen dos tipos de aplicaciones: IVR y WEB que tienen algunas clases en común.

La IVR refleja la interfaz vocal implementada, mientras que la WEB consiste en el desarrollo de una página web para facilitar el mantenimiento de la base de datos por parte de los administradores y los empleados de los cines.

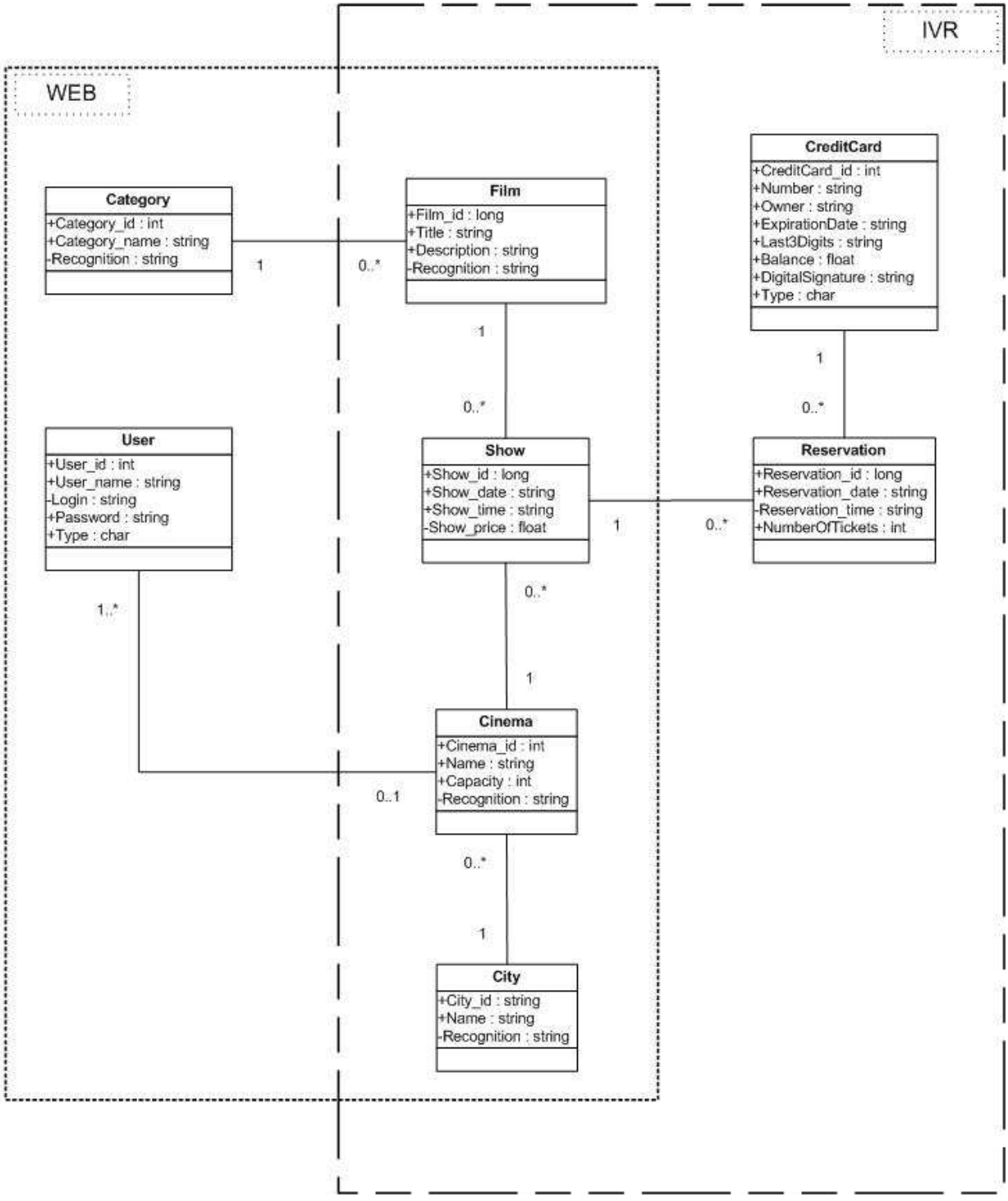


Fig.1 Diagrama de clases

- ***City :***

Esta clase representa la cuidad donde están los cines que hay que gestionar. Tiene los siguientes atributos:

City_id: funciona como identificador de la clase City, o mejor dicho, como clave primaria.

Name: permite almacenar el nombre de la cuidad.

Recognition: como su nombre indica, este campo contiene el conjunto de frases o palabras que el usuario podrá decir a la hora de indicar el nombre de la cuidad de la cual hay que hacer la búsqueda y que serán reconocidas por parte del sistema.

- ***Cinema :***

Esta clase representa los cines de una determinada cuidad. Sus atributos son:

Cinema_id: la clave primaria de la clase Cinema.

Name: para almacenar el nombre del cine.

Capacity: para almacenar la capacidad del cine.

Recognition: para almacenar el conjunto de frases o palabras relacionadas con el nombre del cine, que podrán ser reconocidas por el sistema.

- ***Show :***

Para representar los distintos horarios o turnos de una película en un determinado cine. Como atributos, tiene los siguientes:

Show_id: la clave primaria de la clase Show.

Show_date: este campo es para almacenar la fecha de emisión de una película.

Show_time: este atributo nos permite almacenar la hora del turno de emisión de una película.

Show_price: indica el precio de un billete de un turno determinado.

- ***Film :***

Esta clase representa las películas que se emiten en un determinado cine. Sus atributos son:

Film_id: la clave primaria de la clase.

Title: el título de la película.

Recognition: el conjunto de frases, que representan el título, pronunciadas por parte del usuario y reconocidas por el sistema.

Description: este campo representa una pequeña descripción de la película y un resumen de la misma.

- ***Category :***

Esta clase simboliza las distintas categorías que pueden tener las películas.

Category_id: funciona como clave primaria.

Category_name: expresa el nombre de la categoría.

Recognition: indica lo mismo que los campos de ***Recognition*** anteriormente mencionados, éste está relacionado con el atributo ***Category_name***.

- ***User :***

Representa los administradores encargados del mantenimiento de la base de datos y también los empleados de los cines.

User_id: actúa como identificador de la clase.

User_name: almacena el nombre del administrador.

Login: indica el nombre de usuario para acceder a la página web.

Password: almacena la contraseña del usuario para acceder a la página web.

Type: este campo indica el tipo de privilegio de los diferentes usuarios que acceden a la página web.

- ***CreditCard :***

Esta clase nos permite almacenar los datos bancarios asociados a una tarjeta bancaria determinada.

CreditCard_id: el identificador de la clase.

Number: representa el número de la tarjeta de crédito.

Owner: indica el nombre del titular de la tarjeta bancaria.

ExpirationDate : fecha de caducidad.

Last3Digits : indica el número de verificación de la tarjeta de crédito. Corresponde a los tres últimos dígitos que se encuentran impresos sobre la banda de firma, situada en el reverso de la tarjeta de crédito. Así para proporcionar más seguridad y proteger al titular contra fraudes.

Balance: para almacenar el saldo de la cuenta bancaria.

DigitalSignature: corresponde a la firma digital asociada a la tarjeta de crédito en cuestión, también para ofrecer la máxima seguridad contra fraudes.

Type: indica el tipo de la tarjeta de crédito.

- ***Reservation:***

Toda la información relacionada con una determinada reserva, la almacenamos en esta clase. Sus atributos son:

Reservation_id: la clave primaria de la clase.

Reservation_date: la fecha de la realización de la reserva.

Reservation_time: la hora de la realización de la reserva.

NumberOfTickets: el número de asientos reservados asociados a una reserva.

Las clases ***User, Category, Film, Show, Cinema y City*** son los elementos que forman la página web. Así que el administrador puede realizar las siguientes acciones identificandose previamente:

- dar de alta a una nueva película.
- dar de alta a un nuevo cine.
- dar de alta a un nuevo turno.
- modificar una película que ya existe en la base de datos.
- modificar un cine que ya está dado de alta en la base de datos.
- modificar un turno.
- borrar una película.
- borrar un cine.
- borrar un turno.

Por otro lado, la interfaz vocal incluye todas las clases menos la de ***User.***

3.2. Diagrama de flujo :

Es una técnica utilizada para representar esquemáticamente bien sea la secuencia de instrucciones de un algoritmo o los pasos de un proceso. En nuestro caso es para representar los distintos pasos de las llamadas al sistema. Estos diagramas facilitan la representación de cantidades considerables de información en un formato gráfico sencillo.
Adicionalmente, los diagramas de flujo facilitan a otras personas la comprensión de la secuencia lógica de la solución planteada.

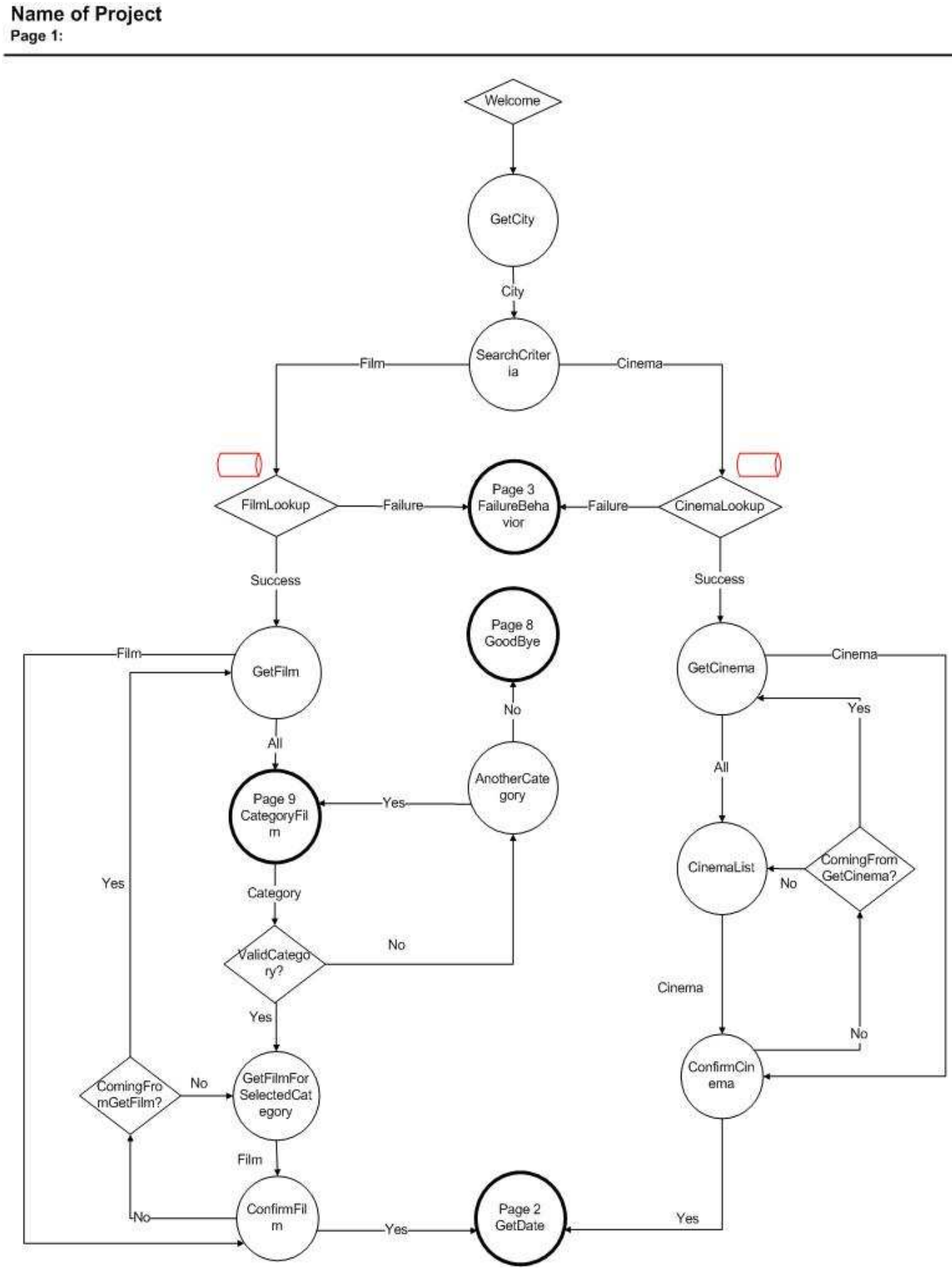


Fig.2 Diagrama de Flujo: página de inicio

La [Fig.2] corresponde a la página de inicio del diagrama de flujo. Como podemos observar se trata de un conjunto de estados y transiciones.
Cuando un usuario llama al sistema el primer estado que se accede es “Welcome” donde se produce un mensaje de bienvenida y se transita al siguiente estado que corresponde al de “GetCity” donde el usuario debería elegir una ciudad para poder continuar. Una vez seleccionada la ciudad, el sistema le pregunta al usuario por el patrón de búsqueda y como ilustra la figura, hay dos opciones: o bien buscar por nombre de película o bien por nombre de cine.

En los estados “FilmLookup” y “CinemaLookup” se accede a la base de datos con el fin de recuperar la lista de películas y de cines respectivamente. Si se ocurre un error, se transita al estado “FailureBehavior”, y en caso contrario se pasa a los estados “GetFilm” y “GetCinema”, respectivamente.

En “GetFilm” y “GetCinema” el usuario tiene dos alternativas, si conoce el nombre de la película o el nombre del cine, lo puede facilitar al sistema, si no, tiene la posibilidad de escuchar toda la lista de todas las películas o los cines almacenados en la base de datos.

En el caso de que el usuario opte por la opción de escuchar toda la lista de películas, se le pregunta por la categoría de películas que le interesa, para poder hacerle escuchar solamente las de la categoría seleccionada. Esto se hace en el estado “GetFilmForSelectedCategory”. Y cuando ya está escogida la película, se transita al estado “ConfirmFilm” para confirmar el nombre de la película, y así evitar arrastrar cualquier error a los siguientes estados.
Lo mismo ocurre con el nombre del cine.
Tras haber confirmado el nombre de la película o del cine, se pasa al estado “GetDate” para poder elegir la fecha de la reserva.

Name of Project
Page 2:

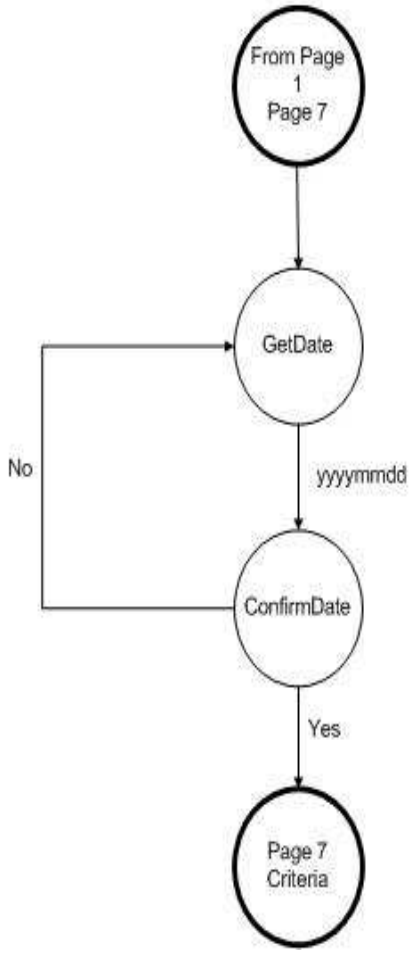


Fig.3 Diagrama de Flujo: Obtener la fecha de reserva

Esta parte del diagrama corresponde a la de validar los datos de entrada asociados a la fecha seleccionada por el usuario. Como vemos en la [Fig.3], primero se le pregunta al usuario la fecha, luego se confirma para validarla. Si se trata de una fecha válida, y tras confirmarla, se pasa al estado siguiente "Criteria", en caso contrario, se vuelve a preguntársela al usuario una vez más.

Name of Project
Page 3:

Fig.4 Diagrama de Flujo: En caso de error

En cuanto a las situaciones de error, se transita al estado “FailureBehavior” en el cual se produce un mensaje avisando al usuario de lo ocurrido, y luego se pasa al estado final correspondiente a “GoodBye”, como está representado en la [Fig.4].

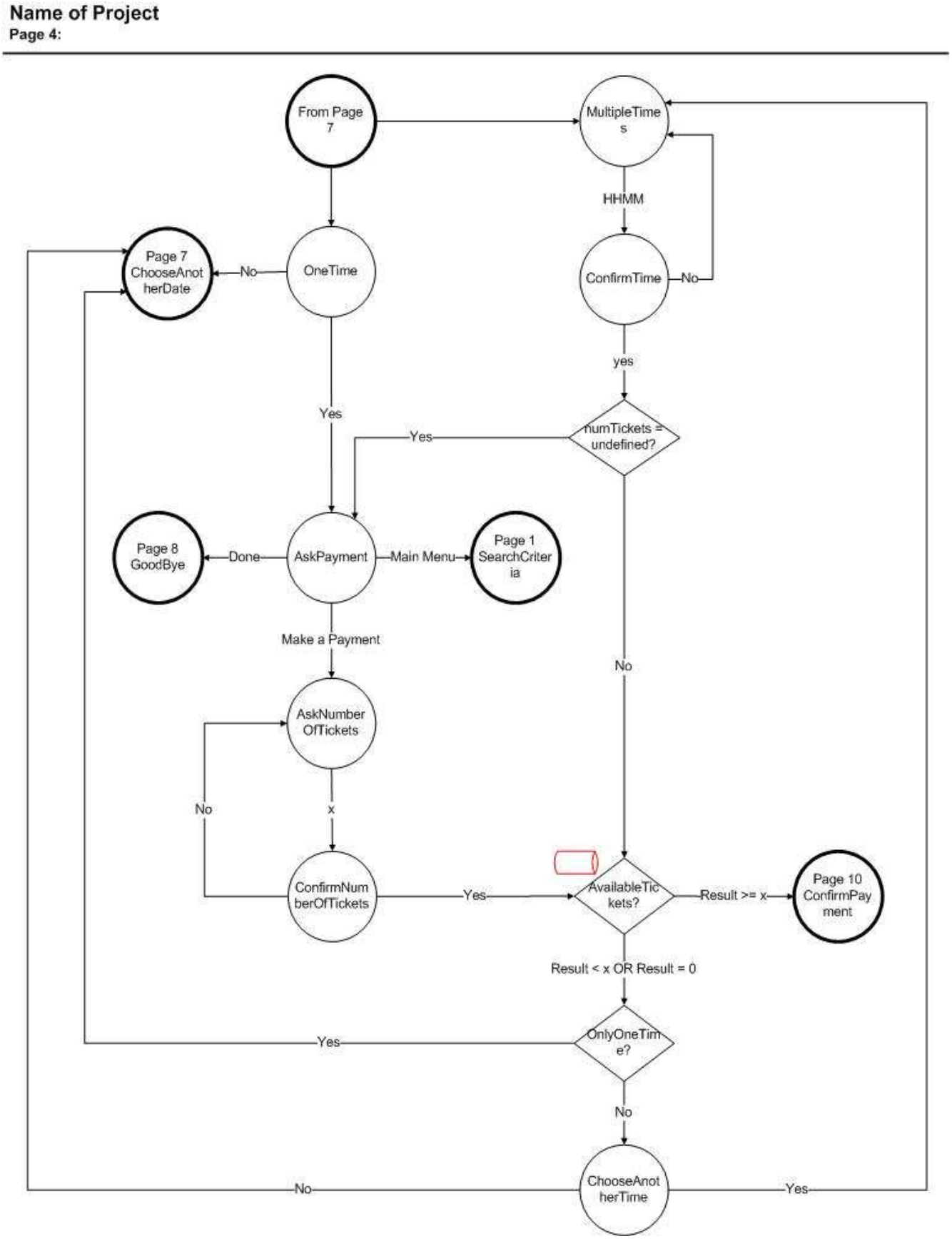


Fig.5 Diagrama de Flujo: Obtener la hora de reserva y realizar el pago

En esta figura [Fig.5] está reflejado el proceso de especificar la hora del turno por parte del usuario y el proceso del pago en el caso de que el usuario esté interesado en hacer la reserva.

Debe observarse que existen dos caminos distintos. El primero corresponde a la situación donde exista solamente un turno en la fecha seleccionada. Esto se ve en el estado “OneTime”. En tal caso el usuario puede aceptar o rechazar la hora hallada, si la acepta, y a continuación se va al estado de “AskPayment”, en el cual tiene tres alternativas, volver al menú principal, salir de la aplicación o bien empezar el proceso del pago. En este último caso, se transita al estado “AskNumberOfTickets” para pedirle el número de asientos a reservar. Después de confirmar este número se hace una consulta a la base de datos para averiguar la disponibilidad de asientos en la fecha especificada, lo cual da lugar a dos posibles caminos. Una posibilidad es que el número de asientos a reservar supere lo que queda por reservar, si esto ocurre, el sistema da al usuario la posibilidad de elegir otra fecha. Sin embargo, si el número de asientos a reservar está por debajo del número total de asientos disponibles, se pasa al estado “ConfirmPayment” para comenzar el procedimiento del pago.

En el otro caso, de que el usuario rechaza la hora encontrada, el sistema le lleva al estado “ChooseAnotherDate” para poder escoger otra fecha.

Por otra parte, en el otro camino que describe la situacion de muchos turnos, sucede lo mismo que en la otra alternativa previamente mencionada, con la diferencia de que el usuario tiene la posibilidad de elegir un turno entre muchos, y se transita al estado “ChooseAnotherDate” solamente si ninguno de los turnos hallados conviene al usuario.

Name of Project
Page 5:

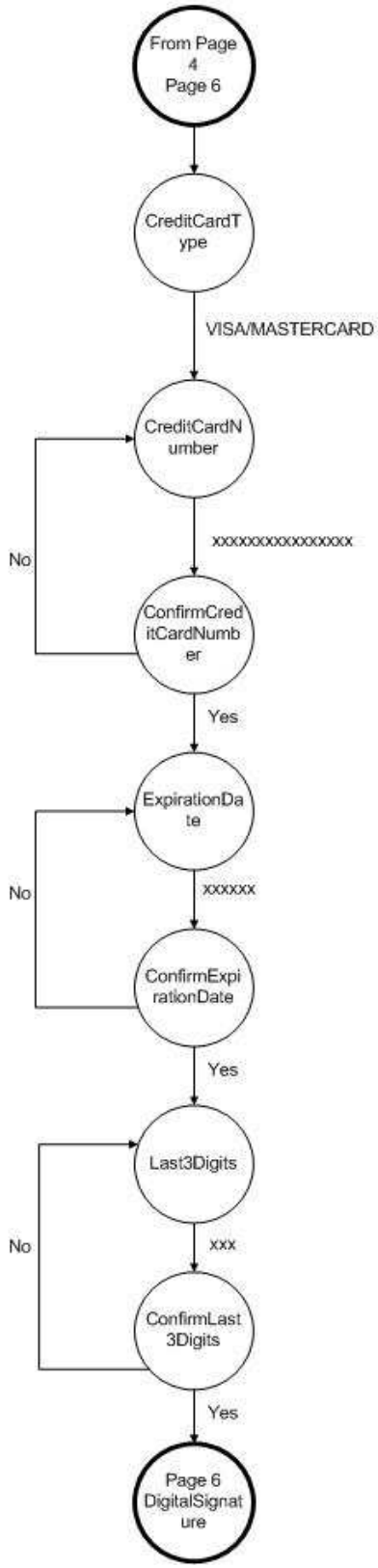


Fig.6 Diagrama de Flujo: Introducir los datos de la tarjeta bancaria

Como su nombre indica, la figura [Fig.6] describe los estados correspondientes al procedimiento de introducción de los datos bancarios por parte del usuario. Obviamente, los datos que se introducen son: el tipo de la tarjeta de crédito, el número de la misma, la fecha de caducidad, los tres últimos dígitos asociados a la misma, y por último, y tras confirmar estos datos, se transita al estado donde se pide la firma digital.

Si observamos, en todas las figuras anteriormente explicadas relacionadas con el diagrama de flujo, existe un círculo que lleva la palabra “from” el cual indica la página desde donde se ha transitado.

Name of Project
Page 6:

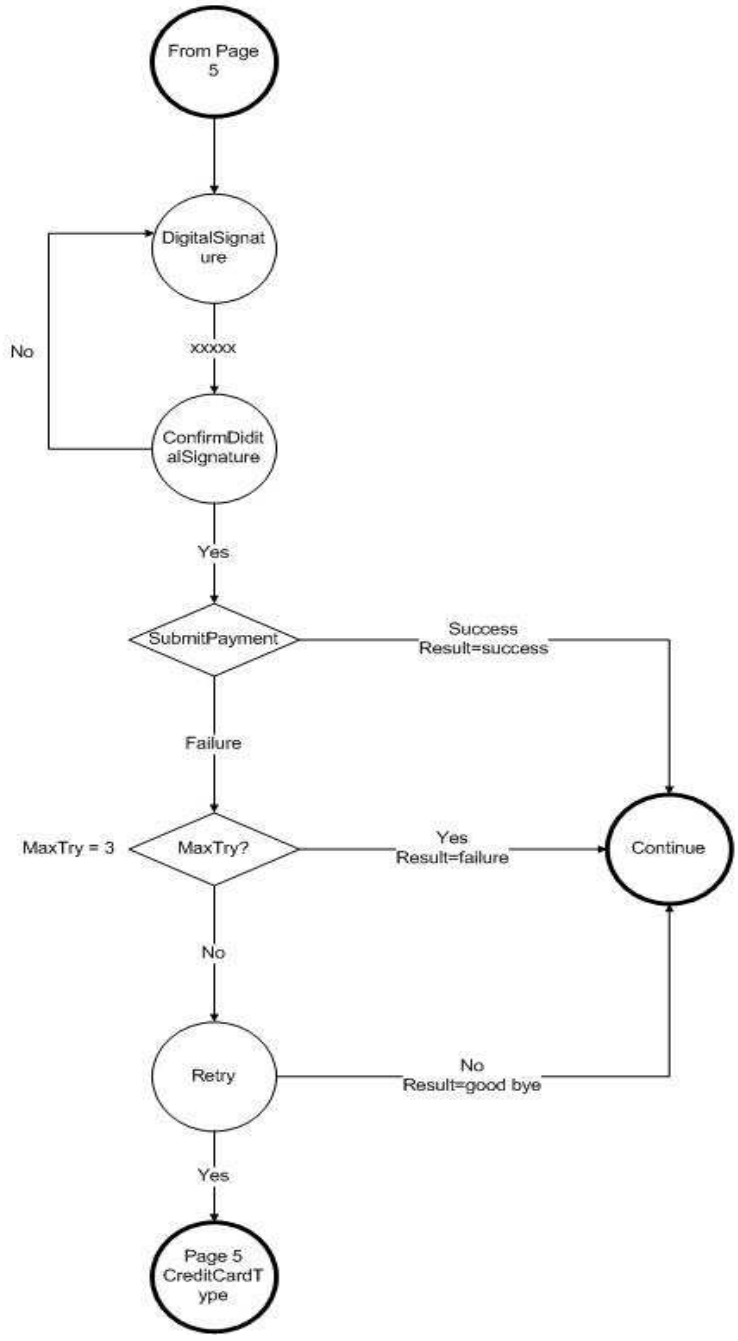


Fig.7 Diagrama de Flujo: Introducir los datos de la tarjeta bancaria(2)

De igual forma, se introduce el resto de datos bancarios, la firma digital, y se confirma igual que los casos anteriores [Fig.7].

Es importante destacar que en caso de introducir datos erróneos el sistema no deja al usuario hacer más de tres intentos con la misma tarejeta bancaria.
Esto es lo que está representado en la [Fig.7], donde podemos observar que se verifica si se cumple la condición “el número de intentos supera 3”, en el punto de decisión “MaxTry”.

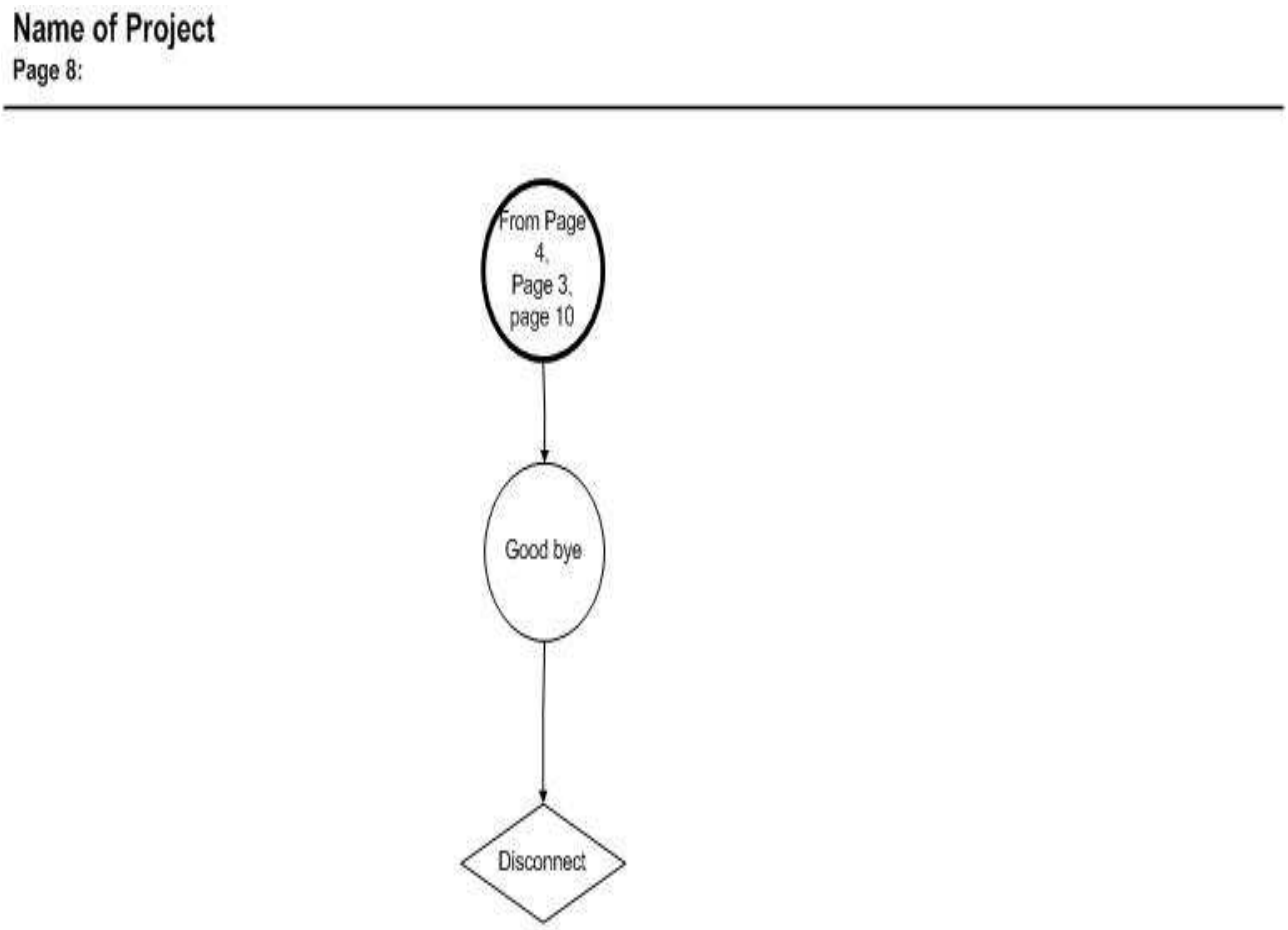


Fig.8 Diagrama de Flujo: Estado final, el final de la llamada

Esta figura [Fig.8] representa el estado final del diagrama de flujo, en el cual se produce el mensaje de despedida, se cuelga y se sale de la aplicación.

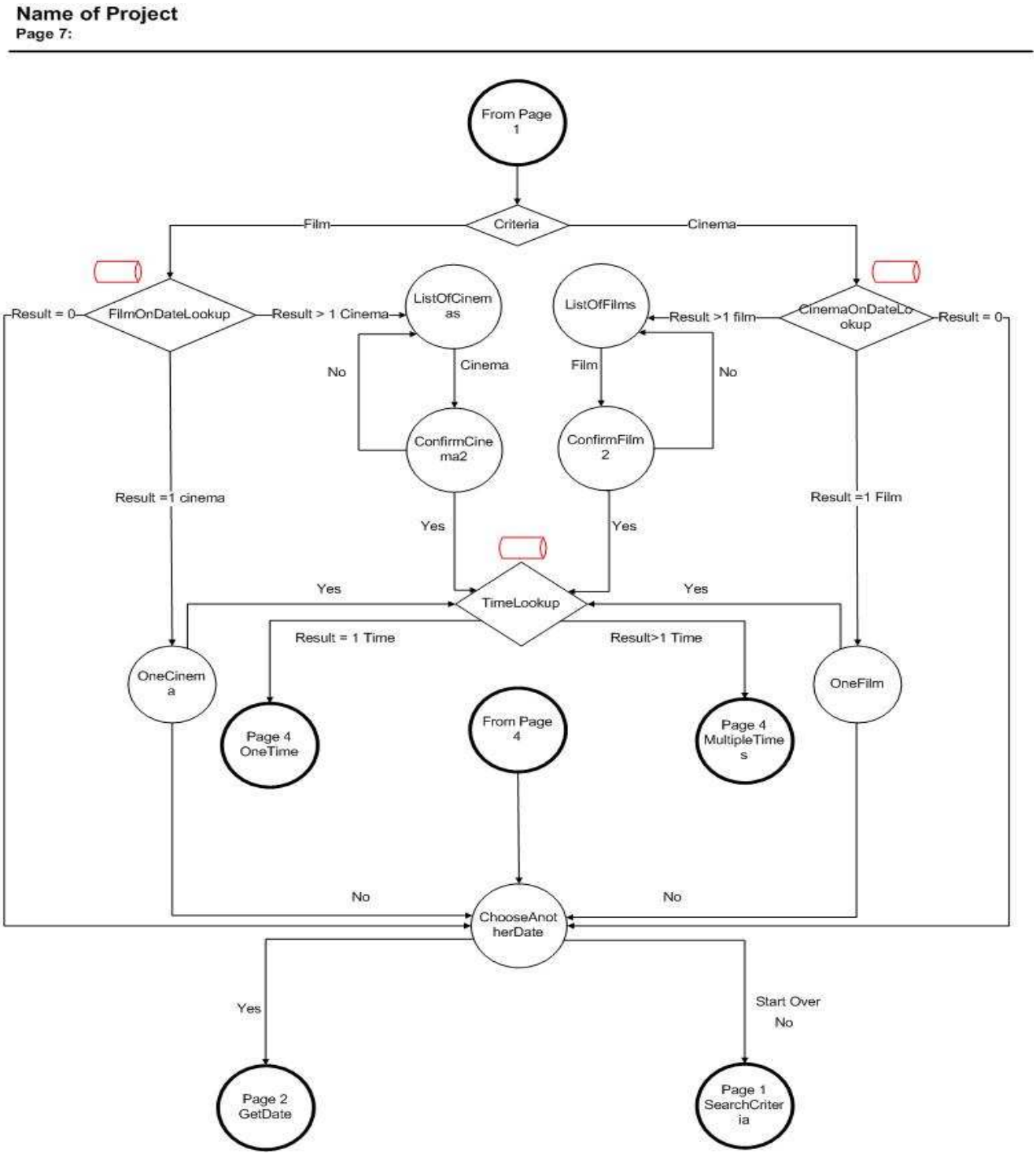


Fig.9 Diagrama de Flujo: Acceso a la base de datos con las preferencias del ususario

Llegamos a la parte del diagrama de flujo donde se accede a la base de datos con las preferencias del usuario:

- Si el usuario había escogido la opción de buscar por nombre de película, entonces nos vamos por la rama de “criteria = film”, llegar a este punto significa que ya tenemos seleccionada la película, la fecha, por lo cual, accedemos con estos datos para recuperar la lista de los nombres de cines que emiten dicha película en la fecha elegida. En la [Fig.9], podemos ver lo que acabamos de mencionar claramente en el estado “FilmOnDateLookup”. Esta consulta a la base de datos, da lugar a tres posibles bifurcaciones. La primera coincide con el caso de que no exista ningún cine que expone la película escogida en la fecha seleccionada, lo cual hace que el sistema conceda al usuario la posibilidad de seleccionar otra fecha.

  En segundo lugar, nos encontramos con el caso de que haya un solo cine, en este caso, el usuario puede aceptar o no el resultado obtenido, transitandose a los estados “TimeLookup” (para elegir un turno) y “ChooseAnotherDate” (para seleccionar otra fecha), respectivamente.

  Finalmente, queda por explicar el caso de que obtengamos una lista de más de un cine, entonces, el usuario escoge uno, y se transita al estado “TimeLookup” para elegir un turno después de haber confirmado el nombre del cine.
- La segunda alternativa, coincide con el hecho de que el usuario había optado por la opción de buscar por nombre de cine, sucede exactamente lo mismo que en el caso anterior, con la única diferencia de que se recupera de la base de datos la lista de películas que se emiten en el cine y la fecha seleccionados.

Name of Project
Page 9:

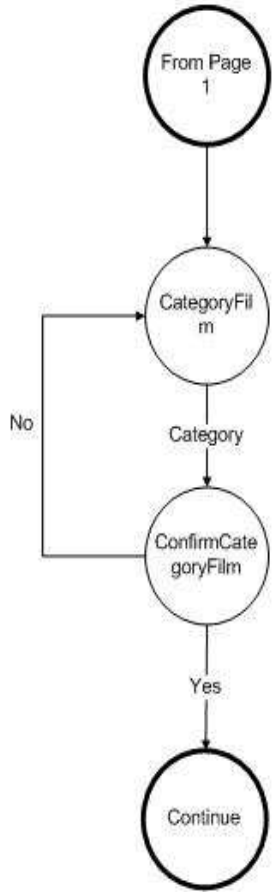


Fig.10 Diagrama de Flujo: Seleccionar una categoría de películas y confirmarla

Veamos ahora la descripción de los estados de elección de la categoría de la película. Como se refleja en la [Fig.10], se le pregunta al usuario una categoría, y luego se confirma.

Name of Project
Page 10:

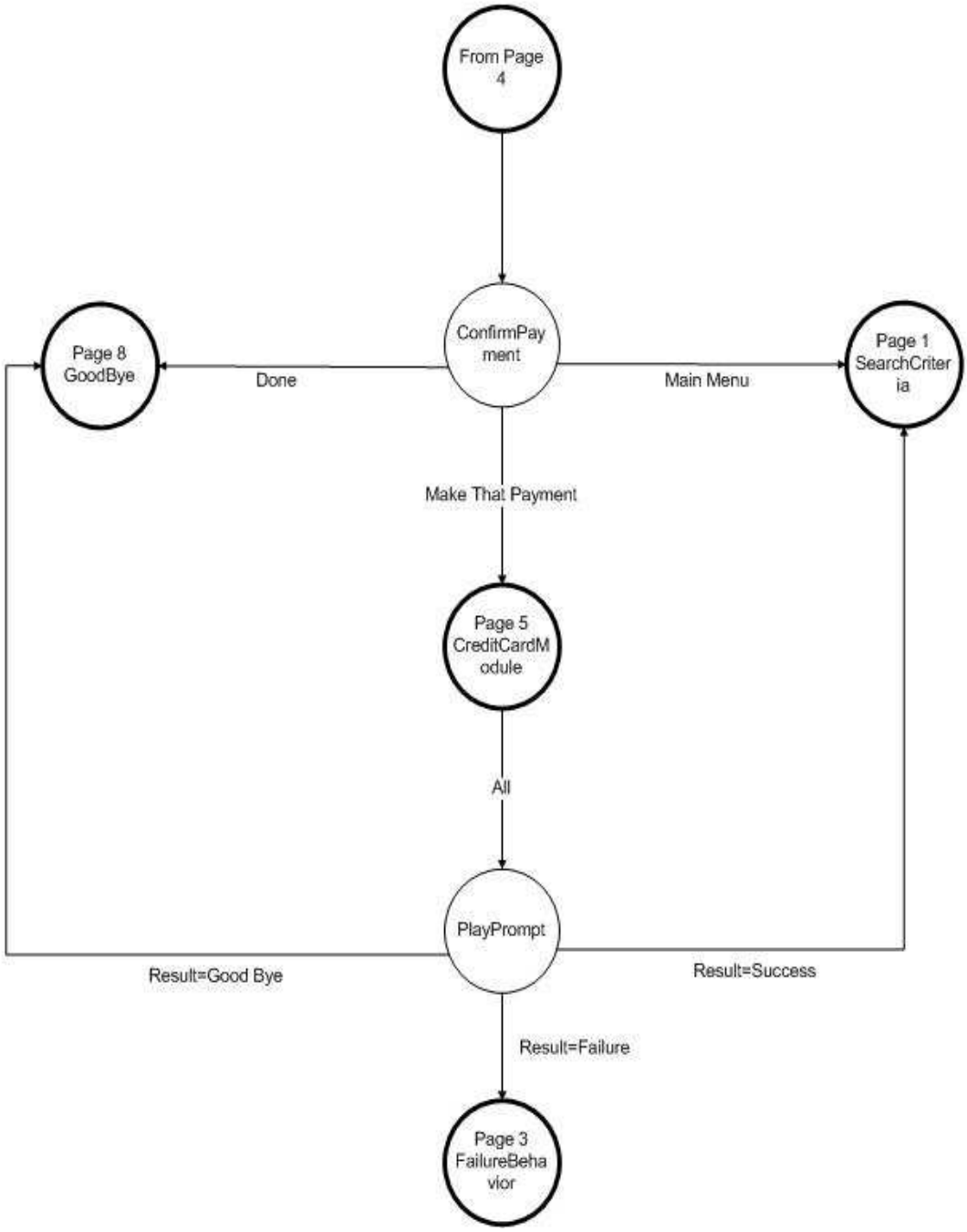


Fig.11 Diagrama de Flujo: Confirmación del pago

Por último, presentamos los estados correspondientes a la confirmación del pago. Como observamos, en el estado “ConfirmPayment”, el usuario puede continuar para efectuar la reserva, volver al menú principal para realizar otra búsqueda o bien salir de la aplicación.

Confirmar el pago lleva al usuario al estado “CreditCardModule” que incluye todos los estados que hemos visto antes, asociados con la introducción de los datos bancarios.

Si fijamos en la figura [Fig.11], hay un estado llamado “PlayPrompt” que es accesible desde el último estado del “CreditCardModule” en el cual, se reproduce un mensaje para recordar el usuario la información de la reserva, como por ejemplo el nombre de la película seleccionada, el número de asientos reservados, el total recaudado de la cuenta bancaria,etc,...

Desde este estado “PlayPrompt”, tenemos tres caminos diferentes, uno correspondiente al caso de éxito cuando se hace la reserva correctamente (lleva la etiqueta “result=succes”) y luego se transita al menú principal para realizar otra acción, el segundo con la etiqueta “result=failure”, éste ocurre cuando ha habido un error a la hora de realizar la reserva y se ha intentado las veces suficientes permitidas por el sistema, de este estado con estas condiciones se pasa al estado “FailureBehavior”. Y el último, corresponde al camino etiquetado como “result=GoodBye” en el cual el usuario no ha superado el número de intentos permitidos, y se transita al estado “GoodBye”.

3.3. Diseño de la base de datos :

3.3.1. Alojamiento de la base de datos :

Para que mi aplicación pueda funcionar debería alojarla en un servidor que soporte: ***java*** y ***mysql***.

Después de haber buscado por internet encontré uno de alojamiento gratis que cumple los requisitos, que permite crear cuentas, de forma totalmente gratuita, para disfrutar de los servicios que ofrece.

El servidor era : http://www.eatj.com/ que tiene muchas ventajas. Creando una cuenta, se puede tener 50 MB de espacio. También, después de crear la base de datos se puede guardar como fichero extensión “sql” para, en el caso de pérdida de datos, solo con la opción de importar, poder crearla nuevamente, en pocos segundos, a partir del fichero previamente mencionado.

3.3.2. Estructura de la base de datos :

Ahora veamos cómo está organizada nuestra base de datos con sus tablas y sus atributos.

La [Fig.12] representa la página principal de nuestro servidor correspondiente a la sección de administrar la base de datos.

La base de datos lleva el nombre “seventhart” y tiene asociadas ocho tablas, como se indica en la figura [Fig.13].

A continuación, representamos figuras que muestran todas las tablas reflejando tanto los atributos como los datos.

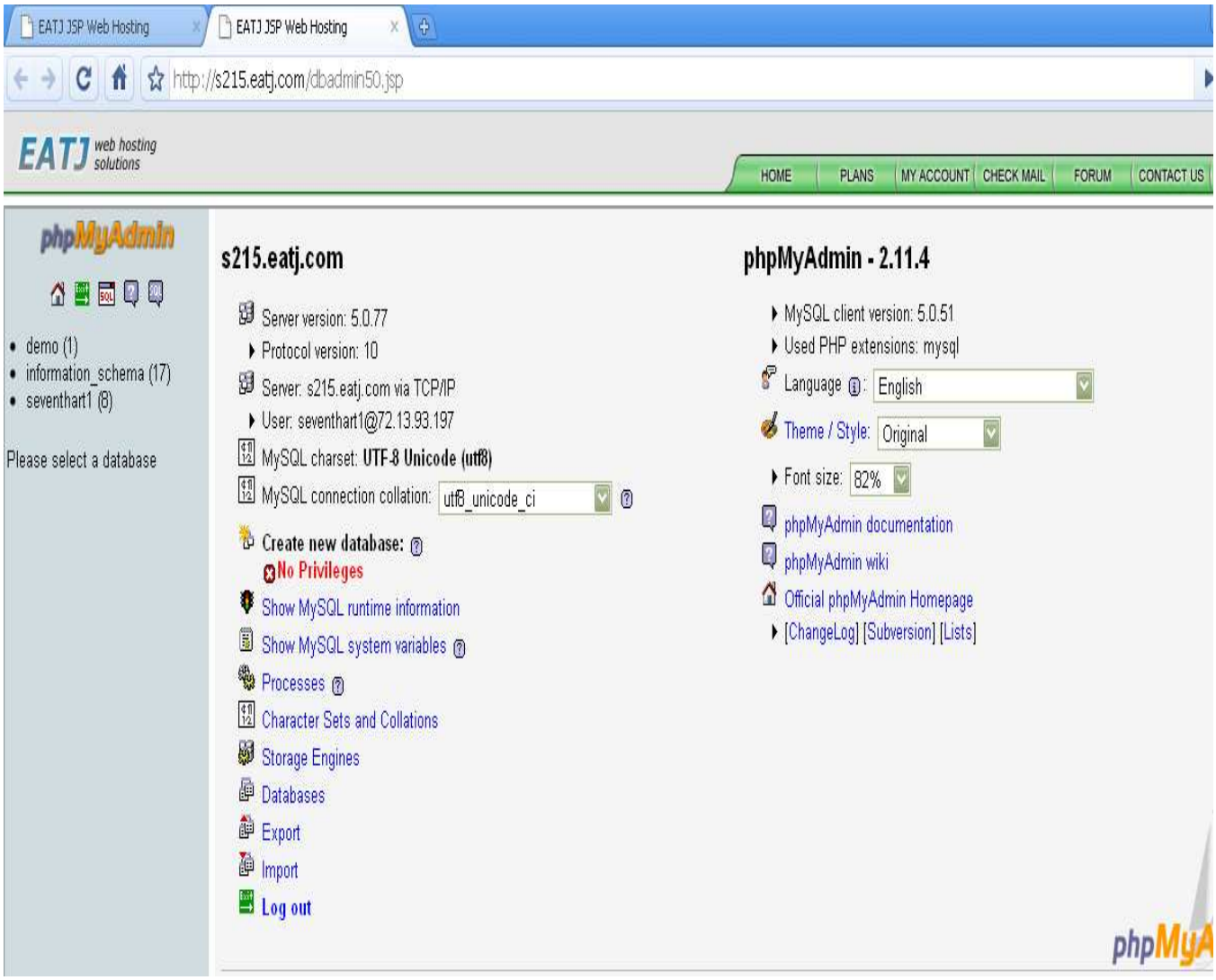


Fig.12 Servidor de alojamiento de la base de datos: página principal

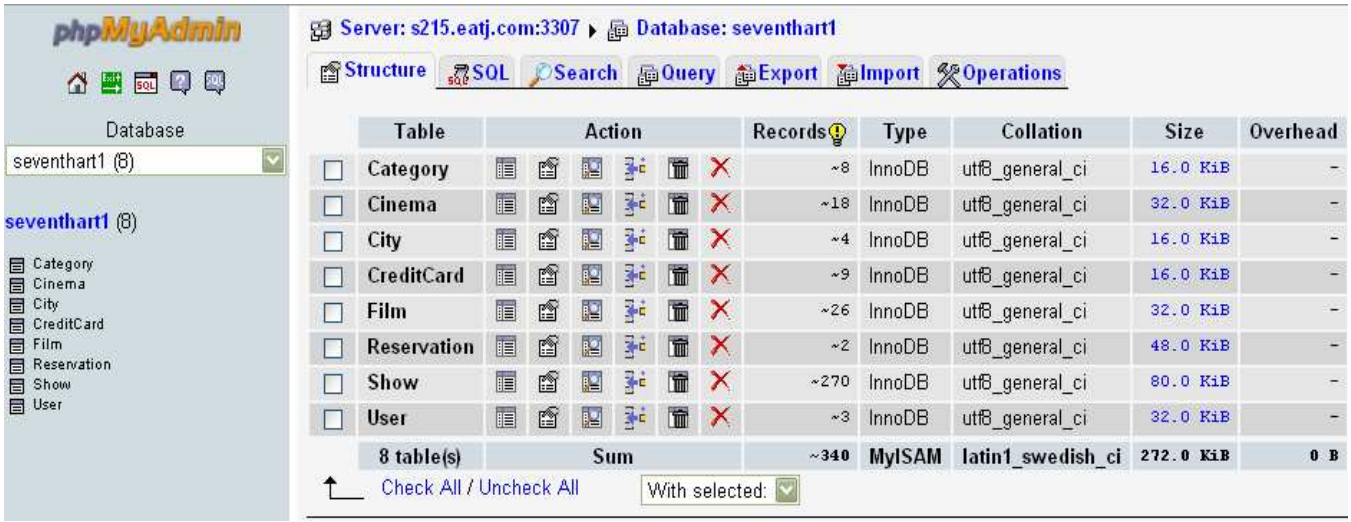


Fig.13 Las tablas de la base de datos.

	Field	Type	Collation	Attributes	Null	Default	Extra	Action
☐	Category_id	int(11)			No		auto_increment	
☐	Category_name	varchar(15)	utf8_general_ci		No			
☐	Recognition	varchar(50)	utf8_general_ci		No			

Fig.14 Tabla "Category": atributos.

	Category_id	Category_name	Recognition
☐	1	Comedy	[comedy comedia]
☐	2	Action	[action accion (dtmf-2)]
☐	3	Romance	[romance (?cuento ?de amor) (love ?story)]
☐	4	Crime	[crime crimen]
☐	5	Adventure	[adventure aventura]
☐	6	Thriller	[thriller (?novela ?de intriga)]
☐	7	Horror	[horror terror miedo]
☐	8	Drama	[drama]

Fig.15 Tabla "Category": datos.

	Field	Type	Collation	Attributes	Null	Default	Extra	Action
☐	Cinema_id	int(11)			No		auto_increment	
☐	Name	varchar(30)	utf8_general_ci		No			
☐	Capacity	int(11)			No			
☐	Recognition	varchar(100)	utf8_general_ci		No			
☐	City_id	varchar(3)	utf8_general_ci		No			

Fig.16 Tabla "Cinema": atributos.

Cinema_id	Name	Capacity	Recognition	City_id
1	Cinesa Bonaire	50	(?cinesa bonaire)	VLC
2	El punt de la Ribera	60	[(el punt ?de ?la ribera) ribera]	VLC
3	Abaco Cullera	66	(abaco cullera)	VLC
4	ABC Gandia	80	(a b c gandia)	VLC
5	Yelmo Cines El Teler	90	(yelmo ?cines ?el teler)	VLC
6	Kinepolis Paterna	40	(kinepolis ?paterna)	VLC
7	ABS El Saler Multicines	77	(?(a b s) ?el saler ?multicines)	VLC

Fig.17 Tabla "Cinema": algunos datos.

Field	Type	Collation	Attributes	Null	Default	Extra	Action
City_id	varchar(3)	utf8_general_ci		No			
Name	varchar(20)	utf8_general_ci		No			
Recognition	varchar(40)	utf8_general_ci		No			

Fig.18 Tabla "City": atributos.

City_id	Name	Recognition
BCL	Barcelona	[barcelona barcelone dtmf-1]
MAD	Madrid	[madrid dtmf-2]
SVQ	Sevilla	[sevilla seville dtmf-3]
VLC	Valencia	[valencia valence dtmf-4]

Fig.19 Tabla "City": datos.

Field	Type	Collation	Attributes	Null	Default	Extra	Action
CreditCard_id	int(11)			No		auto_increment	
Number	varchar(16)	utf8_general_ci		No			
Owner	varchar(30)	utf8_general_ci		No			
ExpirationDate	varchar(6)	utf8_general_ci		No			
Last3Digits	varchar(3)	utf8_general_ci		No			
Balance	float			No			
DigitalSignature	varchar(5)	utf8_general_ci		No			
Type	char(1)	utf8_general_ci		No			

Fig.20 Tabla "CreditCard": atributos.

CreditCard_id	Number	Owner	ExpirationDate	Last3Digits	Balance	DigitalSignature	Type
1	4456342312347890	Antonio Martin	201509	355	6980	mb459	m
2	4567233400001230	Cristine Browne	201503	981	589	23ye5	v
3	8890567854321200	Angelina Foster	201006	983	4098	2yhb8	m
4	3445260619831980	Goerge Murphy	201508	266	8098	9g9hk	m
5	3456879001234000	Sophia Pliger	201610	288	498	4lft6	v
6	1234098978904560	Hillary Duff	201012	266	2.5	472yx	v
7	8923678945632340	Michelle Smith	201111	980	598	2d1ef	v
8	2808198133335660	Taylor Ryan	201005	676	3498	c88cv	v
9	6576456790002332	Youssef Elyalaoui	201903	677	4.35	78572	m

Fig.21 Tabla "CreditCard": datos.

Field	Type	Collation	Attributes	Null	Default	Extra	Action
Film_id	bigint(20)			No		auto_increment	
Title	varchar(30)	utf8_general_ci		No			
Description	varchar(200)	utf8_general_ci		No			
Recognition	varchar(200)	utf8_general_ci		No			
Category_id	int(11)			No			

Fig.22 Tabla "Film": atributos

Film_id	Title	Description	Recognition	Category_id
1	Behind Enemy Lines	Owen Wilson plays a Navy navigator who finds himse...	(?behind enymy lines)	2
2	Flightplan	Jodie Foster is a bereaved women whose daughter di...	(flight plan)	2
3	Tomb Raider	Angelina Jolie plays Adventurer Lara Croft in a r...	(?(lara croft) tomb raide)	2
4	Tears of the Sun	Bruce Willis is a special-ops commander who must d...	(treas of ?the sun)	2
5	Point Break	Keanu Reeves goes undercover as a surf dude to inf...	(point break)	2
6	xXx	Vin Diesel is Xander Cage,an extreme sports athlet...	[xxx (three x)]	2
7	X-Men The Last Stand	Starring Hugh Jachman.When a cure is found to tre...	(x men ?the last stand)	2

Fig.23 Tabla "Film": parte de los datos.

	Field	Type	Collation	Attributes	Null	Default	Extra	Action
	Reservation_id	int(11)			No		auto_increment	
	Reservation_date	varchar(8)	utf8_general_ci		No			
	Reservation_time	varchar(8)	utf8_general_ci		No			
	NumberOfTickets	int(11)			No			
	CreditCard_id	int(11)			No			
	Show_id	bigint(20)			No			

Fig.24 Tabla "Reservation": atributos.

Es importante indicar que en un principio, la tabla "Reservation" se encuentra vacía. Se rellena a medida que se va llamando a la aplicación.

	Field	Type	Collation	Attributes	Null	Default	Extra	Action
	Show_id	bigint(20)			No		auto_increment	
	Show_date	varchar(8)	utf8_general_ci		No			
	Show_time	varchar(4)	utf8_general_ci		No			
	Show_price	float			No			
	Film_id	bigint(20)			No			
	Cinema_id	int(11)			No			

Fig.25 Tabla "Show": atributos.

	Show_id	Show_date	Show_time	Show_price	Film_id	Cinema_id
	1	20100501	1500	6.9	10	1
	2	20100501	1800	6.9	1	1
	3	20100501	2200	6.9	21	1
	4	20100501	1500	6.9	1	2

Fig.26 Tabla "Show": algunos datos.

	Field	Type	Collation	Attributes	Null	Default	Extra	Action
	User_id	int(11)			No		auto_increment	
	User_name	varchar(30)	utf8_general_ci		No			
	Login	varchar(20)	utf8_general_ci		No			
	Password	varchar(20)	utf8_general_ci		No			
	Type	char(1)	utf8_general_ci		No			
	Cinema_id	int(11)			No			

Fig.27 Tabla "User": atributos.

			User_id	User_name	Login	Password	Type	Cinema_id
			1	Patric Brown	patbro	c900934	A	1
			2	Josep Lobato	joslob	z106870	B	1
			3	Dany Freizer	danfreiz	f789456	C	1

. *Fig.28 Tabla "User": datos.*

TERCERA PARTE
FASE DESARROLLO

4. *Fase del desarrollo :*

Resumen: La idea principal de este capítulo consiste en hablar de la implementación de la aplicación desarrollada.
Intentaremos dar una idea de la arquitectura soportada por cualquier aplicación vocal en general y hablar de la tecnología utilizada en nuestro desarrollo tanto las herramientas como los lenguajes.
Finalmente, presentar partes del código con explicaciones y acalaraciones.

Arquitectura VXML :

Antes de pasar a explicar con detalle la parte del desarrollo, me gustaría hablar de la arquitectura soportada por el lenguaje VXML y por cualquier aplicación basada en el procesamiento del habla, e intentar simplificar los componentes que tiene la plataforma soportada por este tipo de aplicaciones vocales y como interactúan con el usuario.

VoiceXML interactúa con el usuario a través de diálogos de voz utilizando componentes de voz que garanticen el reconocimiento de ésta.
La pasarela VoiceXML funciona como un navegador de voz mediante la combinación de reconocimiento automático del habla (ASR), síntesis de voz (Text-To-Speech : TTS), soporte para archivos de audio, reconocimiento de tonos digitales, multifrecuencia de doble tono (DTMF), y las funciones y servicios de telefonía.
El navegador de voz es la aplicación que recibe la solicitud del usuario haciendo ASR, hace la correspondiente solicitud HTTP, y recupera el deseado documento VoiceXML o la correspondiente página Active Server (ASP) o JavaServer Pages (JSP).

El estándar VoiceXML se basa en la arquitectura constituida por los siguientes componentes:

- ***intérprete de VoiceXML:*** (aplicación cliente) procesa los comandos del documento VoiceXML, genera prompts, escucha las respuestas del usuario y ejecuta la lógica de la aplicación.Generalmente realiza consultas a diversos sitios web para obtener la información que se ha de proporcionar al usuario.
- ***servidor de documentos:*** (generalmente un servidor web) procesa las peticiones enviadas por la aplicación cliente y proporciona como respuesta documentos VoiceXML.
- ***entorno del intérprete:*** procesa el documento y responde a las llamadas de los usuarios, monitorizando sus entradas en paralelo con el intérprete; por ejemplo, puede detectar frases específicas que el usuario puede utilizar para solicitar ayuda, cambiar las características de la conversión texto-habla, etc.
- ***plataforma de implementación:*** contiene el hardware telefónico y otros recursos relacionados con el mismo, siendo su misión generar eventos en respuesta a las acciones del usuario (p.e. pulsaciones de botones u órdenes expresadas mediante voz).

VoiceXML es independiente de la plataforma de aplicación en la que una aplicación puede ser desarrollada o desplegada. Esto ofrece flexibilidad a los desarrolladores ya que no se limitan a una plataforma concreta de aplicación.

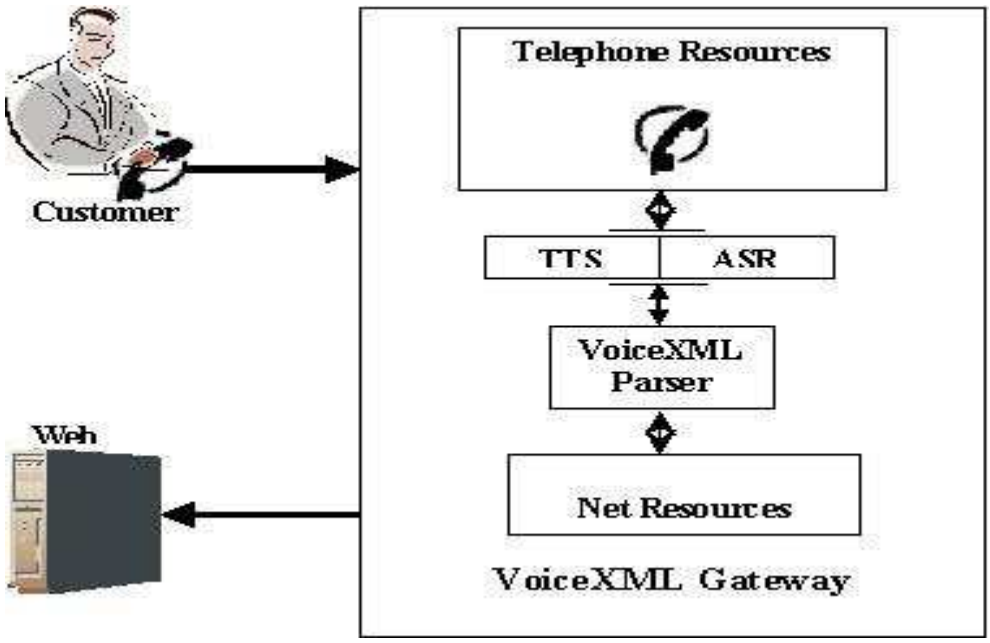


Fig.29 Arquitectura común VoiceXML que muestra los componentes y sus Interacciones.

Es muy costoso y complicado desarrollar una plataforma así para usarla luego en el desarrollo de aplicaciones vocales. Lo ideal sería encontrar una empresa que ofrece la posibilidad de usar sus servicios para probar este tipo de aplicaciones. Hoy en día, hay empresas comerciales que ofrecen portales públicos en los Estados Unidos, tales como BeVocal Café, conocida por este nombre anteriormente y que se ha convertido a Nuance, Estudio de Tellme, y VoiceGenie Taller de Desarrollo, entre otros.

Sin embargo, estos sólo se ofrecen para desarrollar y probar aplicaciones. Con el fin de acceder, se registra con ellos y se obtiene un número de teléfono gratuito y un PIN. Con la finalidad de probar la aplicación VoiceXML, que se aloja en un servidor web cuando se trata de un proyecto enorme, sino, en el mismo servidor de la plataforma si es un fichero VXML de tamaño pequeño, y luego usar el número de teléfono proporcionado para conectar con la pasarela VoiceXML que recupera la aplicación VoiceXML desde el servidor web. Una vez en la fase de producción, se debe comprar un número que puede ser utilizado para fines de comercialización,

o comprar la infraestructura de VoiceXML completa, en cuyo caso deben ofrecer los servicios de telefonía y, posiblemente, VoIP.

La figura [Fig.30] muestra el modelo de arquitectura de VoiceXML en la plataforma de BeVocal (o Nuance).Hay tres componentes:

- ***un servidor web :*** puede ser cualquier servidor web en Internet.
- ***el contexto de intérprete VoiceXML:*** contiene el intérprete de VoiceXML, que es responsable de interpretar el código VoiceXML, proporcionar todas las funciones de apoyo que son necesarias para el intérprete, envíar valores de los parámetros para el servidor web como parte de la solicitud y recibir un documento VoiceXML como respuesta.El servidor web recibe peticiones y envía las respuestas de nuevo al intérprete.
- ***la plataforma de aplicación:*** tiene los componentes de infraestructura, tales como un software de reconocimiento de voz, y un motor de síntesis de voz (TTS). Esta plataforma de aplicación es responsable de la realización de reconocimiento de voz, reproducir archivos de audio, y otras funciones de apoyo.

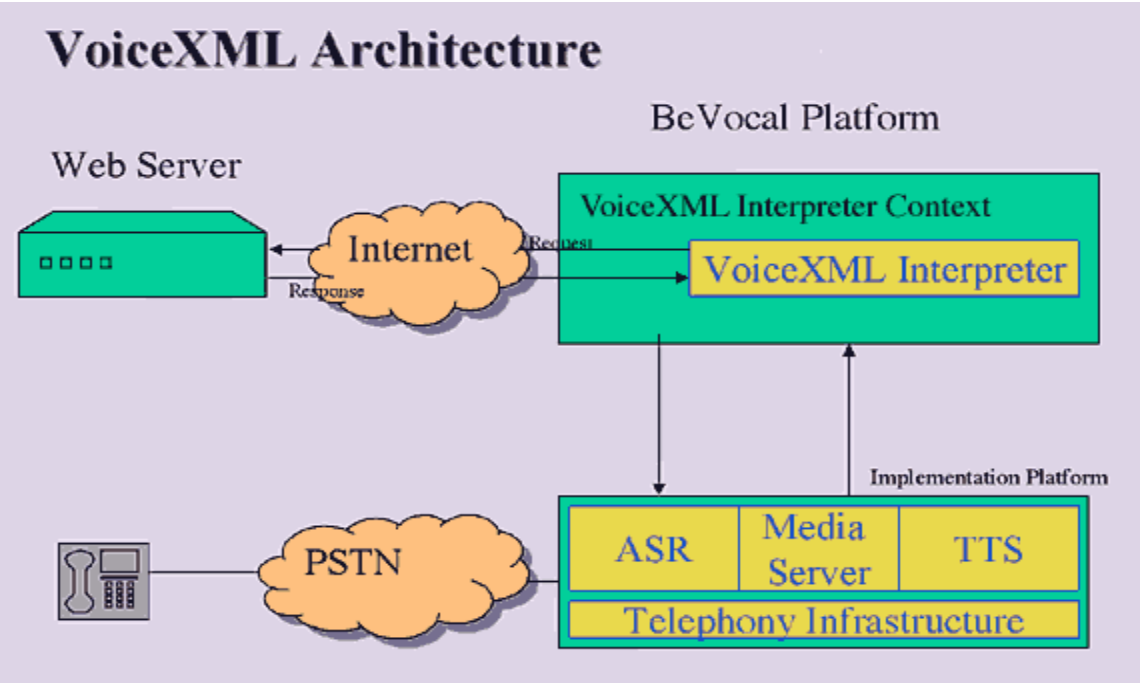


Fig.30 Arquitectura común VoiceXML soportada por Nuance

4.1. Desarrollo de la IVR :

4.1.1. Gramáticas :

Las gramáticas constituyen el mecanismo fundamental de entrada de datos, están escritas según el estándar Speech Recognition Grammar XML Form (GRXML) del W3C, pues deciden qué frases pronunciadas por los usuarios pueden ser reconocidas por parte del sistema.

Para que el reconocedor pueda interpretar la respuesta del usuario es necesario que ésta se corresponda con alguna de las posibles opciones representadas dentro de la gramática.

A continuación, veremos detalladamente los tipos de gramáticas que hemos utilizado para implementar nuestra aplicación.

4.1.1.a. Gramáticas estáticas :

Como su nombre indica, son gramáticas que su contenido no se cambia durante la ejecución de la aplicación.

Mediante este tipo de gramáticas, se pueden especificar un conjunto o set de palabras o frases que el usuario puede decir para llevar a cabo una acción o proporcionar una información, y además se puede proporcionar un valor de retorno para describir la información o la acción solicitada.

Éste es un ejemplo de una gramática estática:

```
REPEAT
[
(?(can you)  repeat ?[that this])
(hear ?that again)
]
```

Teniendo en cuenta que los paréntesis se interpretan como una “AND” lógica, mientras que los corchetes como una “OR” lógica y el punto de interrogación significa que se trata de algo opcional, da igual decirlo o no, podemos concretar el conjunto de frases, que podrían decir los ususarios, relacionado con la gramática anterior.Asimismo, el usaurio puede decir lo siguiente :

- Can you repeat that.
- Can you repeat this.
- Repeat that.
- Repeat this.
- Repeat.
- Hear that again.
- Hear again.

Todas las posibilidades previamente mencionadas son equivalentes.

Veamos ahora otro tipo de gramáticas.

4.1.1.b. Gramáticas DTMF :

Las gramáticas DTMF permiten definir un conjunto de números DTMF que el usuario puede pulsar para llevar a cabo una acción o proporcionar una información, y también permiten proporcionar un valor de retorno para describir la información o la acción solicitada.

Este tipo de gramáticas se usa especialmente en los casos donde se ofrece al usuario la posibilidad de usar la voz o el teclado telefónico para responder a determinadas preguntas.

Consideremos el siguiente ejemplo que ilustra este tipo de gramáticas.

```
MAINMENU
[
(start over)
(main menu)
(dtmf-star)
]
```

Según el ejemplo, da igual que el usuario diga “start over”, “main menu” o pulsar la tecla asterisco del teclado telefónico (ésta última está representada mediante la gramática DTMF) para expresar la acción de volver al menú principal, ya que todas son equivalentes.

La gramática DTMF, se indica mediante la palabra reservada “dtmf” delante del número o símbolo de la tecla en cuestión.

4.1.1.c. Gramáticas dinámicas :

Ahora, explicamos otro tipo de gramáticas que hemos usado para la implementación de nuestra aplicación: el tipo dinámico.

Las gramáticas dinámicas son parecidas a las estáticas, salvo que contienen una parte que está implementada en java (jsp + servlets), cuyo contenido se genera a partir de datos existentes en la base de datos.

Con el uso de esta técnica, la interacción entre la aplicación y el usuario no se ve limitada en un conjunto de frases incambiable, sino, da la posiblidad a la gramática de depender radicalmente de la base de dato, no cabe ninguna duda de que cualquier cambio que afecta a los datos almacenados en la base de datos se verá en la gramática.

Veamos un ejemplo para tener una idea de lo que acabamos de mencionar.

La gramática dinámica “GetFilm.jsp” retornará la lista de todas las películas almacenadas en nuestra base de datos. Una actualización de la base de datos es más que suficiente para que esta gramática devuelva otra lista de películas.

El contenido de la gramática “GetFilm.jsp” es:

```
FILMS
[
 (?I_WANT
[
<%
  String jsonFilms = GrammarUtil.addQuestionMarks(request.getParameter("films"));
  out.println(GetFilm.slots(jsonFilms));
%>
]
?please)
]
```

El resultado de “GetFilm.jsp” después de la ejecución es el siguiente:

```
FILMS
[
(?I_WANT
[
((?(lara croft) tomb raide)) {<choice "Tomb Raider">}
((treas of ?the sun)) {<choice "Tears of the Sun">}
((point break)) {<choice "Point Break">}
 ((?a walk ?to remember)) {<choice "A Walk to Remember">}
 ((addams family ?values)) {<choice "Addams Family Values">}
((domestic disturbance)) {<choice "Domestic Disturbance">}
((scary movie two)) {<choice "Scary Movie 2">}
]
?please)
]
```

4.1.1.d. Gramáticas propias del VXML o gramáticas builtin :

Son gramáticas especialmente diseñadas para las tareas más comunes, y a menudo difíciles, que han sido integradas en el reconocedor como un recurso incorporado.Las gramáticas básicas incorporadas no son solamente la definición de las reglas sino el procesado interno de los resultados.

Los tipos integrados, a veces, proporcionan la entrada tanto en forma de voz como DTMF.

Las builtin especificadas por el estándar de VoiceXML son las siguientes:

- ***Boolean:*** para reconocer respuestas positivas o negativas.

- ***Currency:*** para reconocer cantidades monetarias.Para la entrada de DTMF, la tecla "*" actuará como el punto decimal.
- ***Date:*** se usa este tipo para el reconocimiento de fechas.
- ***Digits:*** permite el reconocimiento de secuencias de dígitos, de longitud limtada o no.
- ***Number:*** para reconocer números.
- ***Phone:*** este tipo integrado permite el reconocimiento de números de teléfono.
- ***Time:*** en esta builtin, se incorporan expresiones de tiempo que especifican horas y minutos.

A continuación, mostramos ejemplos que ilustran la sintaxis y como se incluyen estas builtin en el código.

Ésta, como vemos, corresponde a la builtin booleana, si fijamos bien, observamos que la gramática está parametrizada, se indica la forma de introducir los datos, entonces se asocia la tecla 1 con respuestas afirmativas mientras que la tecla 2 para proporcionar las negativas.

```
<grammar src="builtin:grammar/boolean?y=1;n=2"/>
```

Es importante destacar, que la forma de incluir estas builtin es siempre la misma, es necesario añadir la palabra que indica el tipo para especificar de qué clase se trata.

```
<grammar src="builtin:grammar/date"/>
```

```
<grammar src="builtin:grammar/number"/>
```

```
<grammar src="builtin:grammar/time"/>
```

A veces, es necesario definir parámetros para que la gramática se comporte de la manera requerida.En este ejemplo de la builtin "digits", está indicado que la secuencia debe de ser exactamente de dieciséis dígitos.

```
<grammar src="builtin:grammar/digits?length=16"/>
```

4.1.2. Tecnología utilizada:

En este apartado, describiremos los lenguajes que hemos empleado en el desarrollo de nuestra aplicación. También, hablaremos de las herramientas usadas para esta finalidad.

4.1.2.a. Vxml:

Definición del VXML :

VoiceXML (Voice eXtensible Markup Language) es un lenguaje de programación estándar abierto para el desarrollo de aplicaciones de voz, y no propietario de la familia XML.

A principios de 2001, fue considerado, por el W3C, el lenguaje estándar basado en etiquetas para crear este tipo de sistemas.Poco después, el VBWG (Voice Browser Working Group), un subgrupo de trabajo del W3C dedicado a la evolución del estándar, comenzó a trabajar en la versión 2.0, cuya versión inicial apareció en octubre de 2004.

VoiceXML funciona mediante un navegador de voz cuya salida es audio, y cuya entrada es audio y teclado.La entrada de audio está controlada por un reconocedor de voz integrado con el navegador de VoiceXML.La salida de audio consiste en audio pre-grabado y/o en voz sintetizada por un sistema de Text-To-Speech.

Un sistema VoiceXML para una aplicación concreta consiste en un sistema de documentos, en los que se describen las interacciones con el usuario.El resulatdo de una interacción lleva a la siguiente, por lo que un sistema VoiceXML puede definirse como una máquina de estados finitos.Los documentos VoiceXML incorporan, por cada interacción, los mensajes del sistema, la gramática que

debe utilizarse para reconocer la respuesta del usuario y la siguiente acción a realizar por cada posible respuesta del usuario.

El VoiceXML (o VXML) tiene por objetivo simplificar la creación y el despliegue de servicios vocales por telefonía.

Ventajas del VXML:

Los motivos por los cuales es tan importante la penetración y el enfoque de la tecnología VoiceXML:

En primer lugar, porque el teléfono es una herramienta con mucho poder. Actualmente hay más de 2.2 billones de teléfonos en uso.Una cifra muy superior a la de usuarios que disponen de ordenador con conexión a Internet. Además los teléfonos son fáciles de usar y no necesitan ni tiempo de arranque ni sistemas operativos.

En segundo lugar, porque la voz es importante en el teléfono.La voz siempre ha sido la forma natural de comunicarse a través del teléfono.Aunque algunos móviles disponen de navegadores WAP o XHTML, sus pequeñas pantallas y teclados, hacen complicada la navegación, especialmente en algunas circunstancias como por ejemplo durante la conducción.

Finalmente, VoiceXML es una tecnología independiente de la plataforma, que permite la portabilidad y transferencia de datos entre aplicaciones heterogéneas.

Etiquetas del VoiceXML:

Como ya hemos mencionado antes, el VXML es un lenguaje basado en etiquetas.Una etiqueta es una palabra clave encerrada entre dos corchetes (<y >), que puede tener atributos que van dentro de los corchetes, estos atributos

consisten en nombre y valor, separados por un signo igual (=) y el valor debe escribirse entre comillas.

Cada etiqueta de apertura “<palabra clave>” debe de tener su etiqueta de cierre correspondiente “</palabra clave>”.

Debido a que el VXML incluye varias etiquetas, a continuación, describeremos solamente algunas de ellas, entre otras que hemos usado en nuestra implementación.

- ✓ **<vxml> :**

Todo el contenido de un documento VoiceXML está comprendido entre la <vxml> etiqueta de inicio y la </vxml>etiqueta final.

La etiqueta <vxml> incluye varios atributos, pero los más importantes son :

Versión: indica la versión VXML que ha sido utilizada en este documento.

Xmlns: para especificar el espacio de nombres designado para VXML.Este espacio de nombres se define como http://www.w3.org/2001/vxml.

```
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
.
.
.

</vxml>
```

- ✓ **<form>:**

Como ya hemos explicado antes, un sistema VXML consiste en un conjunto de interacciones con el usuario, o mejor dicho, un conjunto finito de estados.

Cada interacción está contenida entre la etiqueta inicio <form> y la etiqueta final </form>.Hay que tener en cuenta, que cuando se llega al final de un cuadro de dialogo (o form), la ejecución no se pasa al siguiente cuadro de dialogo en el documento, sino, que se indica explícitamente la transición dentro de la interacción actual.

Igual que en el caso anterior, la etiqueta <form> tiene asociada atributos, pero el que hay que indicarlo obligatoriamente es:

Id: especifica el nombre de la interacción.

```
<form id="Welcome">
      <block>
         <audio expr="promptPath('welcome')">
            <value expr="promptValue('welcome')"/>
         </audio>
         <prompt>
            <break time="1s"/>
         </prompt>
         <assign name="context.vars.comingFrom" expr="'Welcome'"/>
         <goto next="#GetCity"/>
      </block>
</form>
```

Este ejemplo, corresponde a la primera interacción que se ejecuta cuando se interactúa con nuestro sistema.La interacción lleva el nombre de “Welcome”.

- ✓ **<audio>:**

Sirve para reproducir un fichero de audio dentro de un sistema.

Como puede verse en el ejemplo adjunto, la etiqueta <audio> consta de un atributo llamado “expr”, el cual indica la ruta donde está almacenado el fichero de audio especificado.

En nuestro ejemplo, el valor de este atributo consiste en una función javascript, “promptPath()”, que devuleve la ruta del archivo de audio llamado “welcome” que le pasamos como parámetro.

Este atributo es solo uno de los varios que incluye esta etiqueta.

```
<audio expr="promptPath('welcome')">
      <value expr="promptValue('welcome')"/>
 </audio>
```

- ✓ **<property>:**

Establece los valores que afectan el comportamiento de la plataforma y contola los ajustes de la misma.

Tiene tres atributos en total, sin embargo, solo es imprescindible indicar los dos siguientes:

Name: para especificar el nomre de la propiedad.El nombre puede ser cualquiera de las propiedades admitidas para el VXML.

Valor: Valor de la propiedad.Los valores permitidos dependen de la propiedad especificada.

Veamos ahora nuestro ejemplo, observamos que se ha especificado la propiedad "bevocal.maxdialogerrors" y se ha establecido el valor 5, esto significa que el número máximo de errores del habla que puede ocurrir dentro de cada interacción no puede superar el valor establecido, en este caso 5.

Se refiere a error del habla, a errores de reconocimiento (que el usuario no ha pronunciado algo bien o lo que ha proporcionado no se ve reflejado en la gramática) y también expiración del tiempo de espera a la hora de esperar una entrada de datos del usuario.

```
<property name="bevocal.maxdialogerrors" value="5" />
```

- ✓ **<script>:**

Permite ejecutar un script en código javascript, o definir funciones en javascript que serán llamadas por expresiones en el mismo ámbito que el elemento que contiene la etiqueta <script>.

Todos los atributos de esta etiqueta son opcionales.En el siguiente ejemplo, no hemos incluido ningún atributo, se ve claramente que se realiza una llamada a una función JavaScript "resetVariables()".

```
<script>
    resetVariables();
</script>
```

- ✓ **<grammar>:**

Esta etiqueta tiene como propósito especificar una gramática que se utiliza en la interacción que la incluye.

La etiqueta <grammar> admite un conjunto de atributos, sin embargo, no vamos a explicar todos de ellos, pero solamente el que aparece en nuestro ejemplo.

Srcexpr: contiene expresión javascript que indica la ruta de la gramática que va a ser utilizada.

En el ejemplo, nuestra expresión javascript, indica la ruta de ubicación de la gramática, "context.path.grammar", concatenada con el nombre del fichero que contiene la gramática "Global.gsl" y indicando el nombre de la gramática en cuestión "#REPEAT".

```
<grammar srcexpr="context.paths.grammar + 'Global.gsl#REPEAT'"/>
```

- ✓ **<foreach>:**

Itera sobre los elementos de un vector.Tiene asociados dos atributos:

Item: suele indicar la variable de iteración, que no puede ser una palabra clave reservada de javascript.

Array: representa el vector sobre el cual iteramos.

El ejemplo muestra una iteración sobre el vector "filmForSelectedCategoryArray", en tal caso la variable de iteración corresponde a "film".

```
<foreach item="film" array="filmsForSelectedCategoryArray">
.....
</foreach>
```

✓ **<subdialog>:**

Invoca a otro diálogo como subdiálogo del actual.

<subdialog> se considera como un mecanismo para la reutilización de los cuadros de diálogo comunes.

El contexto del subdiálogo y del diálogo invocante son independientes, aunque estén en el mismo documento.

El subdiálogo se especifica mediante la referencia indicada en el atributo ***src***.Se pasan parámetros al subdiálogo mediante la inclusión de la etiqueta <param>.

Tiene varios atributos, pero solo comentaremos los siguientes:

Src: especifica la ruta del subdiálogo**.**

Name: el nombre de la variable en la cual el resultado devuelto por el subdiálogo estará almacenado.Para acceder a los valores de retorno del subdiálogo se utiliza la sintaxis: "Name.returnVariableName".

En nuestra aplicación "Seventhart", hemos invocado a un solo subdiálogo que se encarga de realizar el proceso del pago (recogida de los datos bancarios del usuario, realizar el pago y efectuar su reserva).

Este subdiálogo tiene como ruta "CCM.vxml.jsp" y está ubicado en la misma carpeta que el documento principal o invocante, mientras que "CCModule" es el nombre del subdiálogo y de la variable que nos permite acceder a los valores devueltos por éste, efectivamente, "CCModule.result" es el valor devuleto por este subdiálogo.

```
<subdialog name="CCModule" src="CCM.vxml.jsp">
    <param name="subcontext" expr="context"/>
        <filled>
            <log>CreditCardModule subdialog end</log>
            <assign name="context.vars.result" expr="CCModule.result"/>
            <goto next="#PlayPrompt" />
        </filled>
</subdialog>
```

✓ **<data>:**

Esta etiqueta nos permite recuperar o almacenar un dato en un servidor HTTP.Esta operación se realiza sin hacer ninguna transición a otro documento VXML.El dato devuelto por esta etiqueta, es de tipo XML y se guarda en un objeto javascript de solo lectura.

Tiene diversos atributos, describiremos solamente los siguientes:

Name: Nombre de la variable, que debe ser un nombre válido de JavaScript y no puede ser una palabra clave reservada.Si se omite este atributo, la solicitud HTTP se presenta, pero los datos recuperados se ignoran.

Srcexpr: indica la ruta de ubicación del fichero java que recupera el dato en cuestión.

Namelist: este atributo es opcional, y representa una lista de variables separadas por espacio.Estas variables son los parámetros que necesita el fichero java para recuperar el dato en cuestión.

En este ejemplo ilustrativo, se refleja lo que acabamos de mencionar.

```
<data name="mydata" srcexpr="context.paths.data + 'GetCinemasList'" namelist="city"/>
```

4.1.2.b. JavaScript:

JavaScript es el lenguaje que nos permite interactuar con el navegador de manera dinámica y eficaz, proporcionando a las páginas web dinamismo y vida. Se trata de un lenguaje de tipo script compacto, basado en objetos y guiado por eventos, diseñado específicamente para el desarrollo de aplicaciones cliente-servidor dentro del ámbito de Internet.

Los programas JavaScript van incrustados en los documentos HTML, y se encargan de realizar acciones en el cliente, como pueden ser pedir datos, confirmaciones, mostrar mensajes, crear animaciones, comprobar campos...

El VoiceXML utiliza el lenguaje JavaScript para la manipulación de datos, sin embargo, los documentos VoiceXML pueden contener código JavaScript en dos contextos:

- Algunos atributos están en forma de expresiones JavaScript dentro de etiquetas del VXML.

  Veamos el siguiente ejemplo:

```
<assign name="data" expr="1+3" />
```

  Como podemos observar, "1+3" es una expresión JavaScript que asignamos al atributo "expr" de la etiqueta <assign>.

- O bien, cuando se trata de una sección de código JavaScript, lo que solemos hacer, es escribirla dentro de la etiqueta <script> del VXML. Por ejemplo, un script que contiene la función factorial puede ser escrito del siguiente modo:

```
<script>
< ![CDATA[
 function factorial(n){
   return (n <= 1) ? 1 : n*factorial(n-1);
 }
]]>
</script>
```

Ciertos caracteres en JavaScript tienen un significado pero en VXML tienen otro. Por ejemplo, el carácter "<", en JavaScript representa "inferior que", mientras que en VXML, representa el símbolo de apertura de las etiquetas. Lo mismo pasaría para el carácter ">", que en JavaScript indica "superior que" pero se interpreta como cierre de etiquetas en VXML.

Para evitar cualquier error de sintaxis en VXML, cada uno de estos caracteres se representa mediante una secuencia que indica el mismo significado.
Imaginemos que queremos escribir la siguiente expresión JavaScript en nuestro código VXML:

```
balance < minbalance
```

Como ya hemos apuntado, "inferior que" significa otra cosa en VXML. Así pues, la inclusión de esta expresión en código VXML será la siguiente:

```
<if cond="balance &lt; minbalance">
```

En el caso anterior, como se observa, se ha reemplazado el carácter "<" por la secuencia "<".

Por otra parte, cuando incluimos un código JavaScript en la etiqueta <script>, es muy recomendable insertarlo en la sección "CDATA" lo cual nos permite utilizar los caracteres "<", ">" y "&" directamente sin necesidad de reemplazarlos por la secuencia "<", ">" y "&" respectivamente.
La etiqueta <script> también tiene asociado el atributo "***Src***", el cual hace referencia a un fichero JavaScript. Mediante dicho atributo, se indica la ruta de ubicación del fichero en cuestión.
Vista la complejidad de los scripts de este proyecto, hemos empleado esta técnica, la etiqueta <script>, para separar los ficheros VXML de los de JavaScript. De este modo, el código desarrollado queda más claro y legible.

A parte de la propiedad mencionada anteriormente, la etiqueta <script> permite la reutilización del código JavaScript invocándolo desde el fichero VXML. La acción de invocar siempre debe realizarse al inicio del fichero VXML. Esto se observa claramente en el siguiente fragmento del código:

```
<script src="../javascript/global.js"/>
 <script src="../javascript/json.js"/>
 <script src="../javascript/prompts.js"/>
 <script src="../javascript/configuration.js"/>
```

- ✓ **global.js:**

En este fichero, es donde tenemos la declaración de las variables globales que hemos empleado en nuestra aplicación. Estas variables se invocan en cualquier ámbito de la misma.

En este fichero también podemos encontrar la implementación de todas las funciones JavaScript que hemos utilizado para la validación y la manipulación de los datos.

A continuación, mostramos un fragmento de código contenido en dicho fichero.

```
var max_errors = 3;
var confirmConfidence =  0.8;
var maxTry = 3;

function randomize(num) {
  return Math.floor(Math.random() * num);
}
```

Este segmento de código indica la declaración de tres variables globales (max_errors, confirmConfidence y maxTry).

Además, se puede observar la implementación de una función JavaScript "randomize()" que devuelve números aleatoriamente.

- ✓ **json.js:**

Se trata de un fichero JavaScript que viene ya implementado y está disponible en la página oficial de "JSON".

Este fichero contiene funciones en JavaScript que procesan sobre objetos de "JSON".
De dicho documento, la función que nos ha sido de mucha utilidad es "toJSONString(json)", que convierte un objeto "JSON", que le pasamos como parámetro, a una cadena de caracteres.

- ✓ **prompts.js:**

Como ya sabemos, en una interfaz vocal la salida de la voz desempeña un papel importante y esencial para proporcionar información al usuario e interactuar con él (p.e. solicitarle un determinado dato que probablemente haya sido incorrectamente reconocido). Para ello, El lenguaje VXML dispone de un elemento llamado "prompt", el cual controla la salida de la voz sintetizada y de los audios pregrabados. La palabra "prompt" también hace referencia a los mensajes que se reproducen a la hora de interactuar con el usuario.
Aunque el VXML incorpora la técnica de conversión texto-habla, que permite generar oralmente cualquier mensaje en forma de texto mediante un sintetizador de voz, hemos utilizado el mecanismo de mensajes pregrabados ya que permiten que, la interacción con el sistema, parezca más natural.

Por consiguiente, en este fichero "prompt.js" hemos anotado los distintos prompts asociados a cada interacción, incluyendo los casos de error (no se proporciona ninguna entrada o no se reconoce con suficiente confianza la frase pronunciada por parte del usuario). Cada prompt, representa un elemento del objeto JavaScript "PROMPTS" que habíamos declarado en dicho fichero.
Primero se indica al compilador que busque el prompt en forma de audio pregrabado, en el caso de que el prompt no exista, se accede a este objeto, y se busca el prompt en cuestión, que se reproduce en forma de TTS (voz sintetizada).

A modo de ejemplo, mostramos a continuación un fragmento de código extraído de este fichero, que corresponde al prompt de bienvenida que se reproduce en la primea interacción con el sistema.

```
var PROMPTS = {
//welcome :
   "welcome": "Welcome to the seventh art automated system.",
.....
};
```

- ✓ **configuration.js:**

Por último, veamos para que sirve este fichero JavaScript que hemos utilizado en nuestra aplicación.

Vista la complejidad de nuestra aplicación, y la cantidad de ficheros que hemos utilizado para llevar a cabo el desarrollo de la misma. Hemos apuntado en este fichero las rutas de todos los ficheros empleados (gramáticas, prompts, etc.) en variables JavaScript facilitando así su inclusión en el código a la hora de programar y evitando la escritura de la ruta cada vez que se invoca un fichero de datos necesario.

Este fragmento de código, muestra una parte del fichero. Como se puede observar, “context.paths.grammar” es una variable JavaScript, que le hemos asignado la ruta de ubicación del documento donde están almacenadas nuestras gramáticas.

```
context.paths.grammar      = '../grammar/';
```

4.1.2.c. Java y Jsp:

JSP (***JavaServer Pages***) es una tecnología Java que permite generar contenido dinámico para web, en forma de documentos HTML, XML o de otro tipo. Las JSP's permiten la utilización de código Java mediante scripts.

Hemos contado con esta tecnología para implementar ***Servlets*** permitiendo el acceso a la base de datos y la realización de consultas SQL.
Además, la utilización de JSP ha sido en las gramáticas dinámicas que su contenido depende de la base de datos.

4.1.2.d. Json:

JSON, acrónimo de ***JavaScript Object Notation***, es un formato ligero para el intercambio de datos. Es un subconjunto del lenguaje JavaScript que se basa en la construcción de una lista ordenada de valores, listas de objetos, que pueden incluir a su vez tablas hash, objetos con una colección de pares nombre/valor.
La simplicidad de JSON ha dado lugar a la generalización de su uso, especialmente como alternativa a XML.

En nuestra aplicación, los datos que nos devuelven los ***Servlets/Jsp***, vienen en forma de objetos JSON.

Para entender bien lo que proporciona esta tecnología, observemos este ejemplo que corresponde a un objeto JSON cuyo contenido es una lista de cines:

```
{ "status" : "success", "cinemas" : [ {"name" : "Cinesa Bonaire", "recognition" : "(?cinesa bonaire)"}, {"name" : "El punt de la Ribera", "recognition" : "[(el punt ?de ?la ribera) ribera]"},  {"name" : "ABC Gran Turia Multicines", "recognition" : "(?(a b c) ?gran turia ?multicines)"} ] }
```

Este objeto JSON está formado por dos elementos principales:

- El atributo “status”: es una cadena de caracteres cuyo valor nos indica si el acceso a la base de datos ha tenido éxito o fallo.
- El atributo “cinemas”: se trata de un vector que contiene tres elementos (cada elemento está contenido entre dos llaves: {}), cada uno es a su vez un objeto de dos atributos simples que representan: el nombre del cine, recognition respectivamente.

4.2. Desarrollo de la WEB:

4.2.1. WEB : consultas:

En capítulos anteriores hemos mencionado que en nuestra base de datos disponemos de tres tipos de usuarios:

- Tipo C: representa los empleados de las taquillas de los cines. Acceden a la página web para poder mostrar las distintas reservas realizadas para una película en concreto en una fecha dada.
- Tipo B: es el encargado de mantener la base de datos (insertar una nueva película, borrar una película, modificar un turno de emisión, etc.).
- Tipo A: este tipo hace referencia a los administradores de la base de datos. Más adelante veremos las tareas de cada tipo de usuarios.

En este apartado describiremos la página web desarrollada para el tipo C.
Los tres tipos citados anteriormente tienen una página en común donde introducen el nombre de usuario y la contraseña.

Fig.31 Página de identificación.

Dependiendo del tipo se accede a una página o a otra ya que cada usuario tiene una página distinta del resto de los usuarios.

Después de haber introducido los datos de identificación, se validan éstos mediante un ***servlet java*** y se accede a la página de menú asociada al usuario C.

Se trata de una página muy sencilla. Primero, se introduce una fecha, de la que queremos mostrar sus reservas asociadas.

Fig.32 Campo para seleccionar la fecha.

Se da al botón enviar y se muestra una lista de películas emitidas en dicha fecha. En el caso de que no haya ninguna película emitida en esta fecha se muestra un mensaje de aviso.

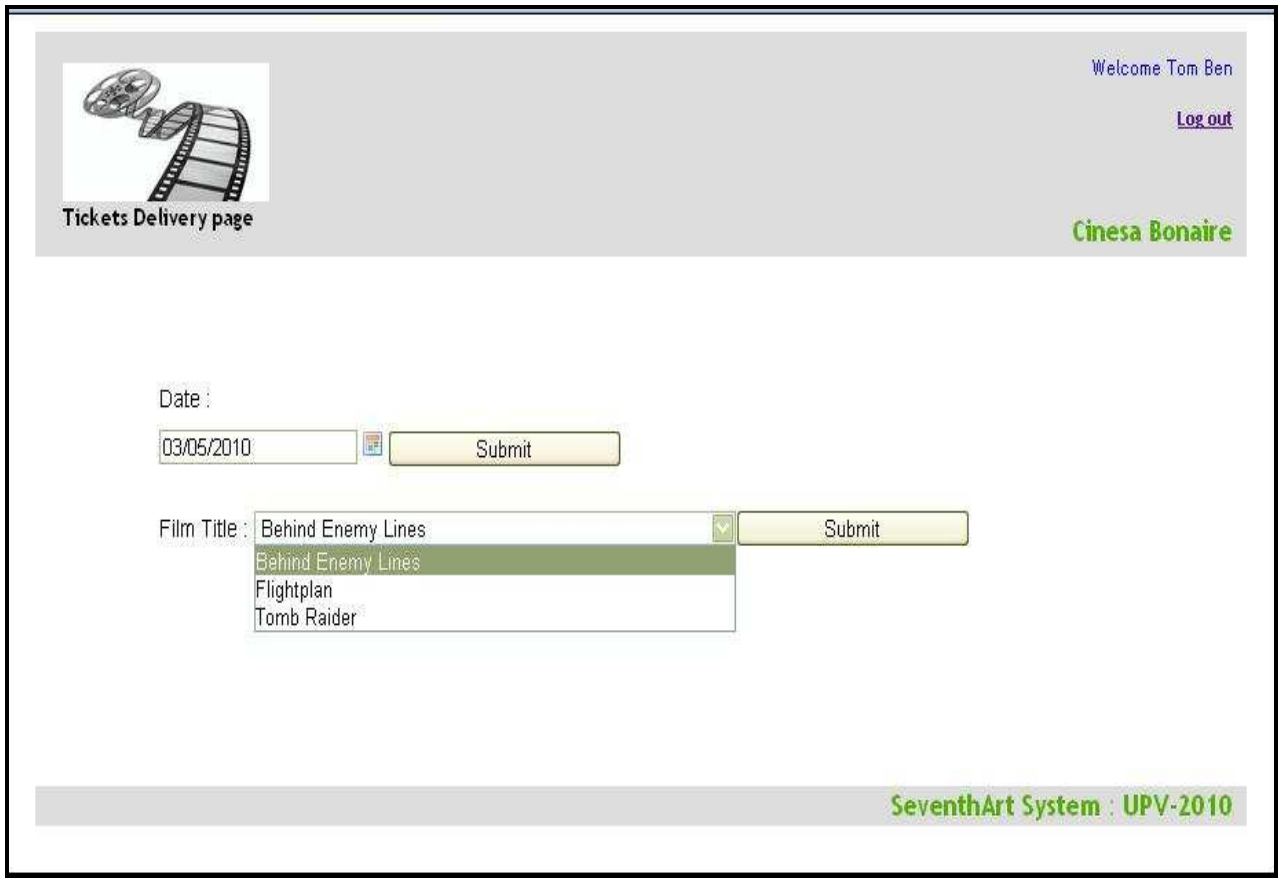


Fig.33 Lista de las películas emitidas en la fecha introducida.

Fig.34 Caso de no existir ninguna película emitida en la fecha seleccionada.

Luego se selecciona la película y se da al botón de aceptar, y se muestra a continuación otra lista que representa los distintos turnos de emisión de la película elegida previamente.

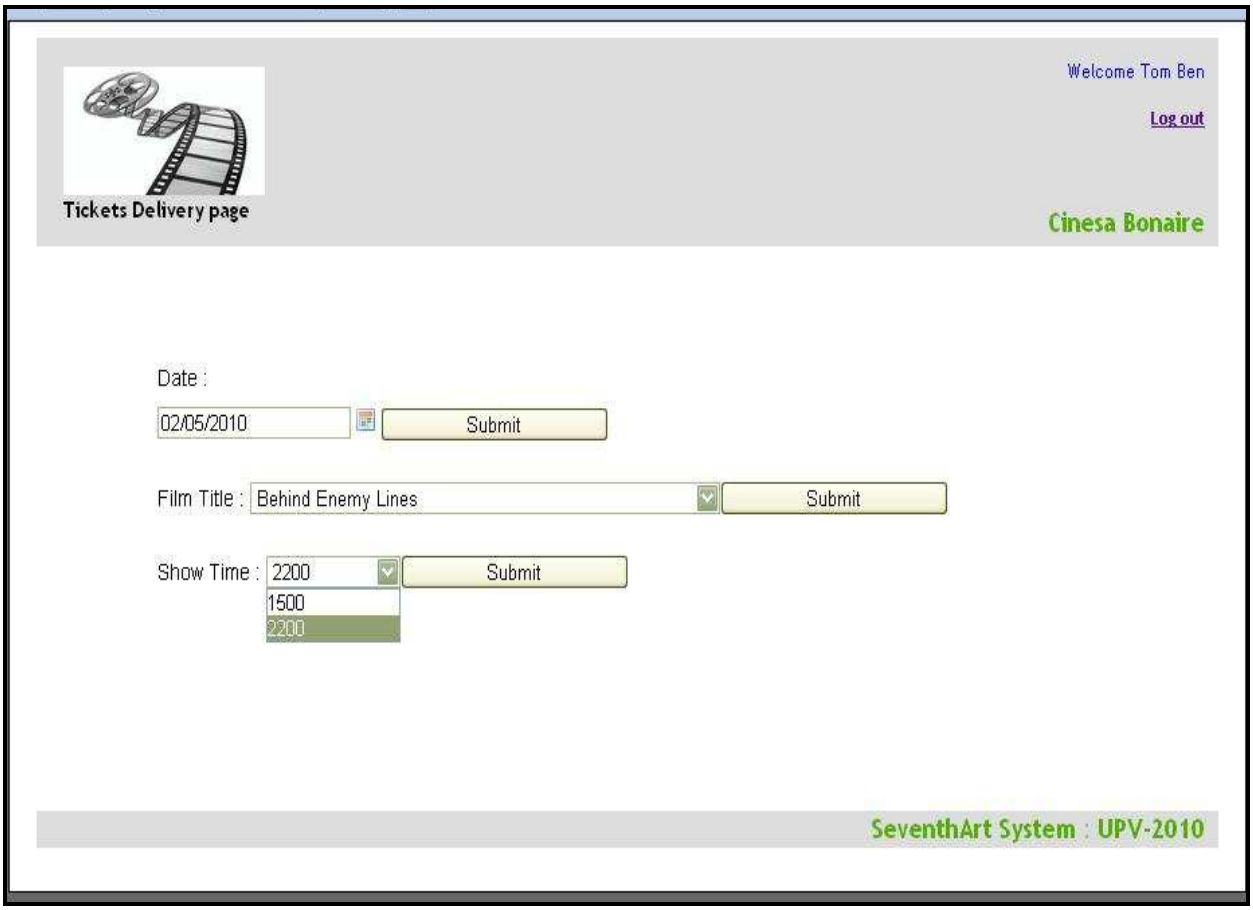


Fig.35 Lista de los turnos de emisión de la película elegida.

Al seleccionar un turno se muestra una tabla con todas las reservas efectuadas para este turno. Los datos mostrados en esta tabla son: el nombre de la persona que realizó la reserva, el número de asientos reservados, etc.

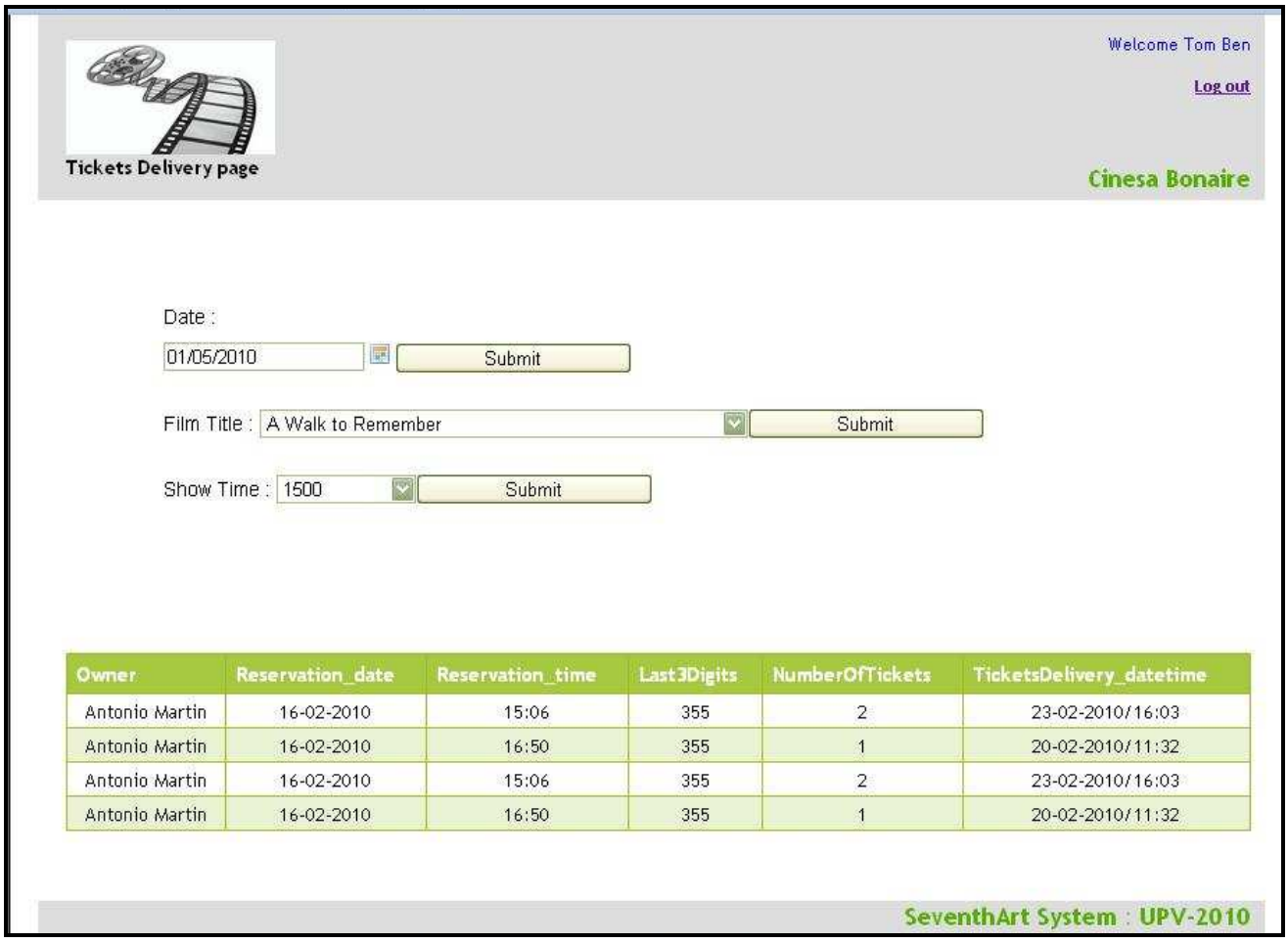


Fig.36 Tabla con todas las reservas efectuadas para la película, el turno y la fecha seleccionados.

En el caso de que no se ha realizado ninguna reserva se muestra un mensaje de error. Ver [Fig.37].

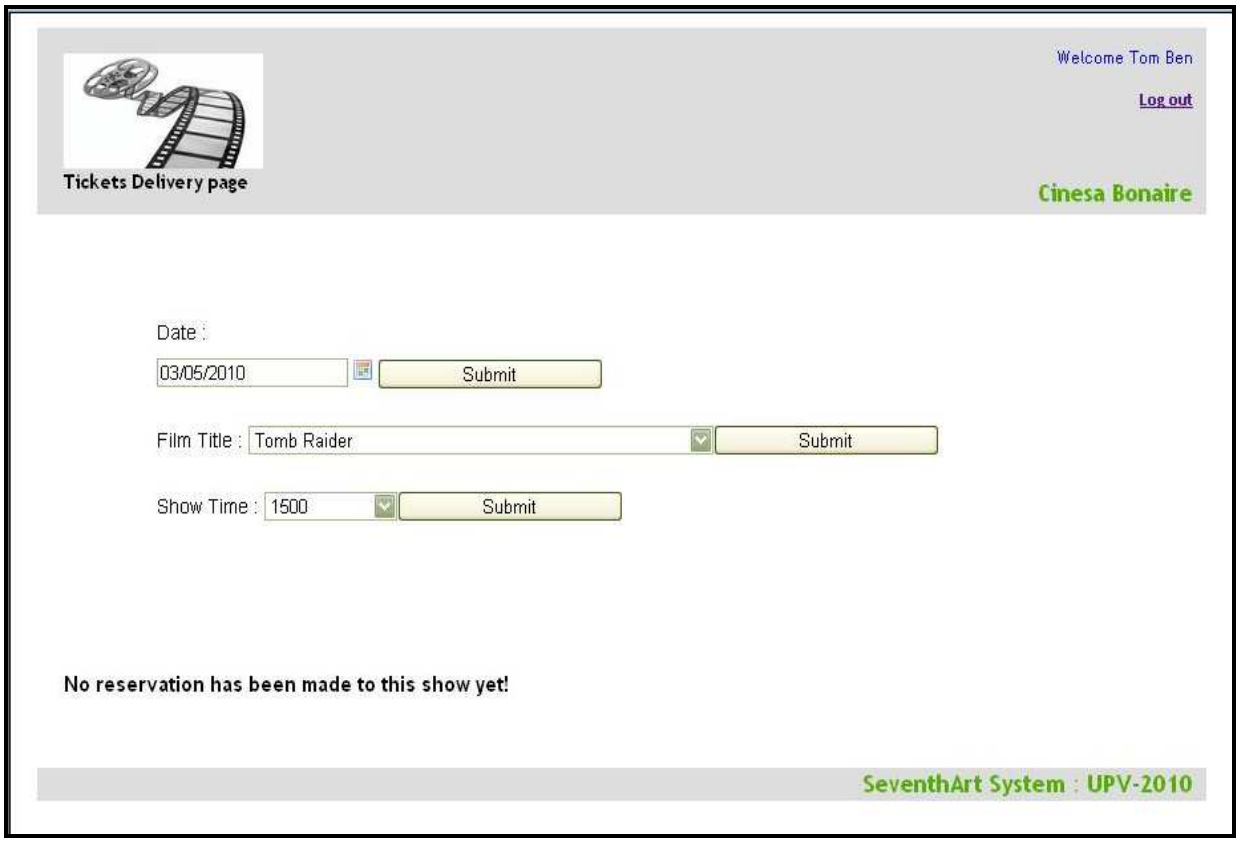


Fig.37 El caso de no existir reservas realizadas.

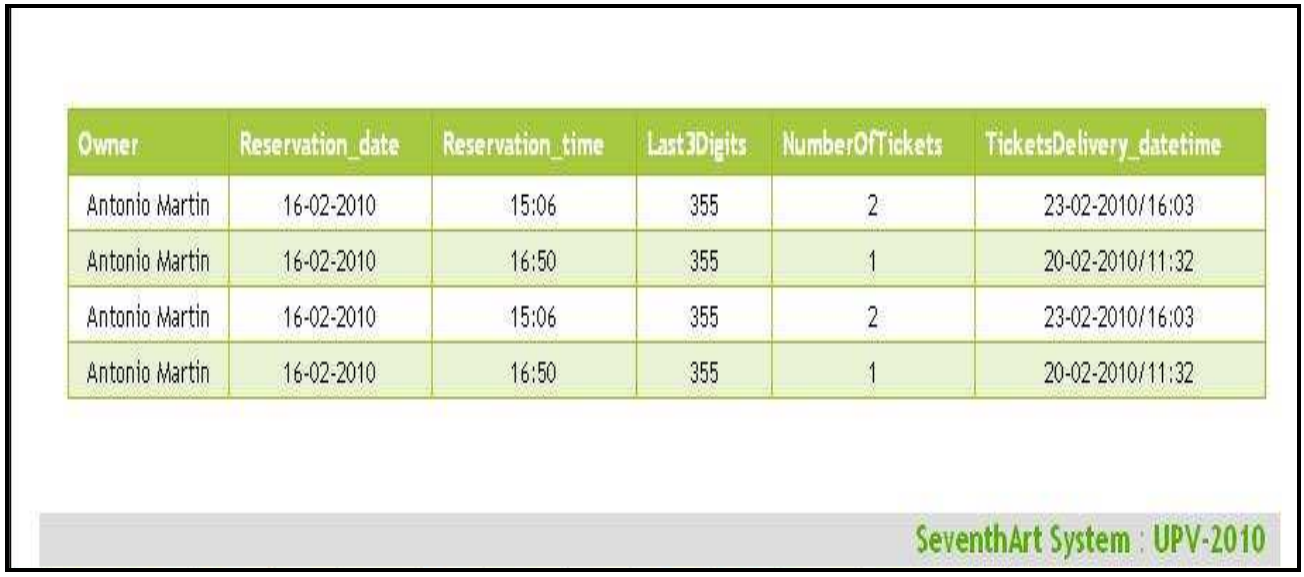
Solo para recordar, las reservas se hacen normalmente mediante la aplicación vocal y la recogida de los tickets se realiza en la taquilla.

Para facilitar la operación de recogida y evitar cualquier situación de error, hemos añadido un nuevo campo en la tabla de ***reservation*** llamado ***"TicketsDelivery_datetime"*** que en principio es una cadena vacía, lo cual significa que no se han recogido aún los tickets.

En otro caso, cuando se han recogido los tickets, se almacena en este campo la fecha de recogida.

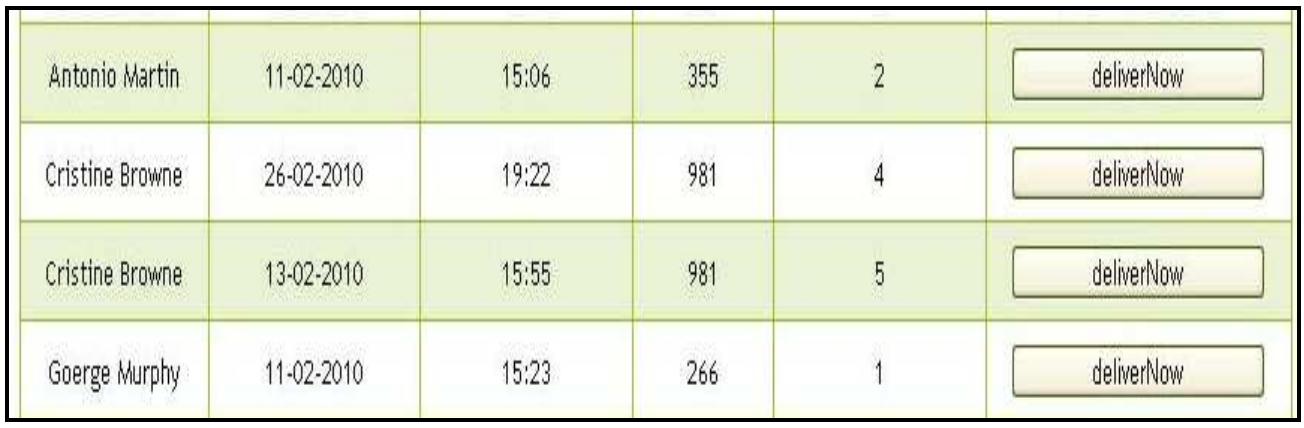
Teniendo en cuenta estos dos casos, el aspecto de la tabla que muestra las reservas efectuadas varía según el caso.

- Aparece una casilla cuyo contenido representa la fecha de recogida [Fig.38].
- Cuando no se han recogido aún, esta casilla aparece como un botón, el cual si se pulsa se ejecuta un ***servlet java*** cuya función es actualizar la base de datos insertando la fecha de recogida.



Owner	Reservation_date	Reservation_time	Last3Digits	NumberOfTickets	TicketsDelivery_datetime
Antonio Martin	16-02-2010	15:06	355	2	23-02-2010/16:03
Antonio Martin	16-02-2010	16:50	355	1	20-02-2010/11:32
Antonio Martin	16-02-2010	15:06	355	2	23-02-2010/16:03
Antonio Martin	16-02-2010	16:50	355	1	20-02-2010/11:32

Fig.38 Reservas con tickets recogidos.



Antonio Martin	11-02-2010	15:06	355	2	deliverNow
Cristine Browne	26-02-2010	19:22	981	4	deliverNow
Cristine Browne	13-02-2010	15:55	981	5	deliverNow
Goerge Murphy	11-02-2010	15:23	266	1	deliverNow

Fig.39 Reservas con tickets pendientes de recoger.

La página incorpora un enlace para permitir al usuario desconectarse en cualquier momento. El desarrollo de estas páginas se ha hecho mediante: ***servlets java, html y jquey***.

En nuestra web se controla la entrada de datos antes de validarlos. Dicho de otra forma, se muestra un mensaje cuando no se introduce el nombre de usuario, la contraseña o ambos en el proceso de identificación.

Fig.40 Mensaje de error cuando no se ha introducido algún campo de identificación.

4.2.2. WEB : insertar datos:

En este capítulo hablaremos de la página web correspondiente a los usuarios de tipo B y A.

Esta parte queda como futuro trabajo y solo hablaremos de la funcionalidad que tendrán estas páginas sin hablar de la implementación.

La página del usuario tipo B incorpora un menú para efectuar las siguientes acciones:

- Dar de alta a una película.
- Dar de alta a un turno de emisión
- Dar de baja a una película
- Dar de baja a un turno de emisión.
- Modificar una película que ya existe en la base de datos.
- Modificar un turno que ya está dado de alta.

El usuario de tipo A o el administrador puede realizar todas las acciones asociadas al usuario de tipo B. Además, puede efectuar las siguientes tareas:

- Dar de alta a un cine.
- Dar de alta a una ciudad.
- Dar de alta a un usuario.
- Dar de baja a un cine.
- Dar de baja a una ciudad.
- Dar de baja a un usuario.
- Modificar un cine.
- Modificar una ciudad.
- Modificar un usuario que ya está dado de alta.
- Asignar un usuario a un cine.

CUARTA PARTE
FASE DE PRUEBAS

5. *Fase de pruebas:*

Resumen: En este capítulo, describiremos el proceso del ***Testing*** (pruebas de software) que hemos utilizado para verificar el funcionamiento de nuestra aplicación, revelar la calidad de la misma y identificar posibles fallos de implementación.
Hablaremos también de la traza de ejecución que nos ha sido de mucha utilidad en la depuración y por consiguiente para descubrir los errores de programación.
Finalmente, presentaremos las herramientas que han sido utilizadas en esta fase para entender mejor el objetivo de este capitulo.

Test y traza de ejecución:

En un proyecto informático, la fase de las pruebas es la fase más relevante en el ciclo de vida del software desarrollado. Ya que gracias a ésta, se puede saber si la aplicación implementada funciona correctamente y si realmente hace lo que se debe hacer y no hace lo que no se debe.

Como ya hemos anotado al principio, nuestra base de datos está alojada en un servidor gratuito que soporta ***Java*** y ***Mysql***. Se trata del servidor www.eatj.com en el cual hemos podido también alojar nuestra aplicación.

En una aplicación vocal, uno de los componentes imprescindibles para el funcionamiento de la misma es la plataforma de implementación.
En capítulos anteriores, hemos comentado que la plataforma de implementación es la responsables de la realización de reconocimiento de voz, reproducir archivos de audio, y otras funciones de apoyo gracias a sus módulos de

infraestructura, tales como un software de reconocimiento de voz y un motor de síntesis de la misma.
Actualmente existen multitud de estas plataformas vocales que permiten a los usuarios acceder a información de todo tipo mediante la interacción con diálogos. Estas plataformas se caracterizan por un alto grado de estandarización e interoperabilidad así como un alto coste.

Teniendo en cuenta que resultaría muy costoso desarrollarlas, para llevar a cabo el funcionamiento de nuestro sistema, nos hemos registrado en la página de Nuance, que dispone del entorno de desarrollo VXML y ofrece una amplia variedad de valiosas herramientas y otros recursos con el objetivo de ayudar a los desarrolladores para construir aplicaciones vocales de alta calidad.
Una vez creada la cuenta, de forma gratuita, teníamos la posibilidad de desplegar nuestra aplicación en la misma página. No obstante, como se trata de un proyecto enorme con diversos ficheros, hemos optado por la otra alternativa que era indicarle al intérprete de la plataforma el enlace de alojamiento de nuestro sistema (el servidor mencionado anteriormente) sin necesidad de desplegar los documentos.

En realidad, estas plataformas vocales actúan como un intérprete y permiten únicamente compilar o interpretar el código desarrollado. Para poder probar este tipo de aplicaciones, hace falta emplear la tecnología VoIP (voz sobre IP) que engloba multitud de tecnología y protocolos de señalización y transporte destinados a hacer posible el establecimiento de llamadas y la transmisión de conversaciones a través de Internet o simplemente realizar llamadas telefónicas mediante un teléfono sea fijo o móvil.

Nosotros hemos optado por la tecnología VoIP mediante la cual podíamos llamar al número gratuito, suministrado por la plataforma con el fin de probar

la aplicación, facilitando el número de identificación y la contraseña proporcionados en el proceso de suscripción, logramos establecer una conexión con nuestro sistema y interactuar con él con la finalidad de verificar el funcionamiento del mismo.

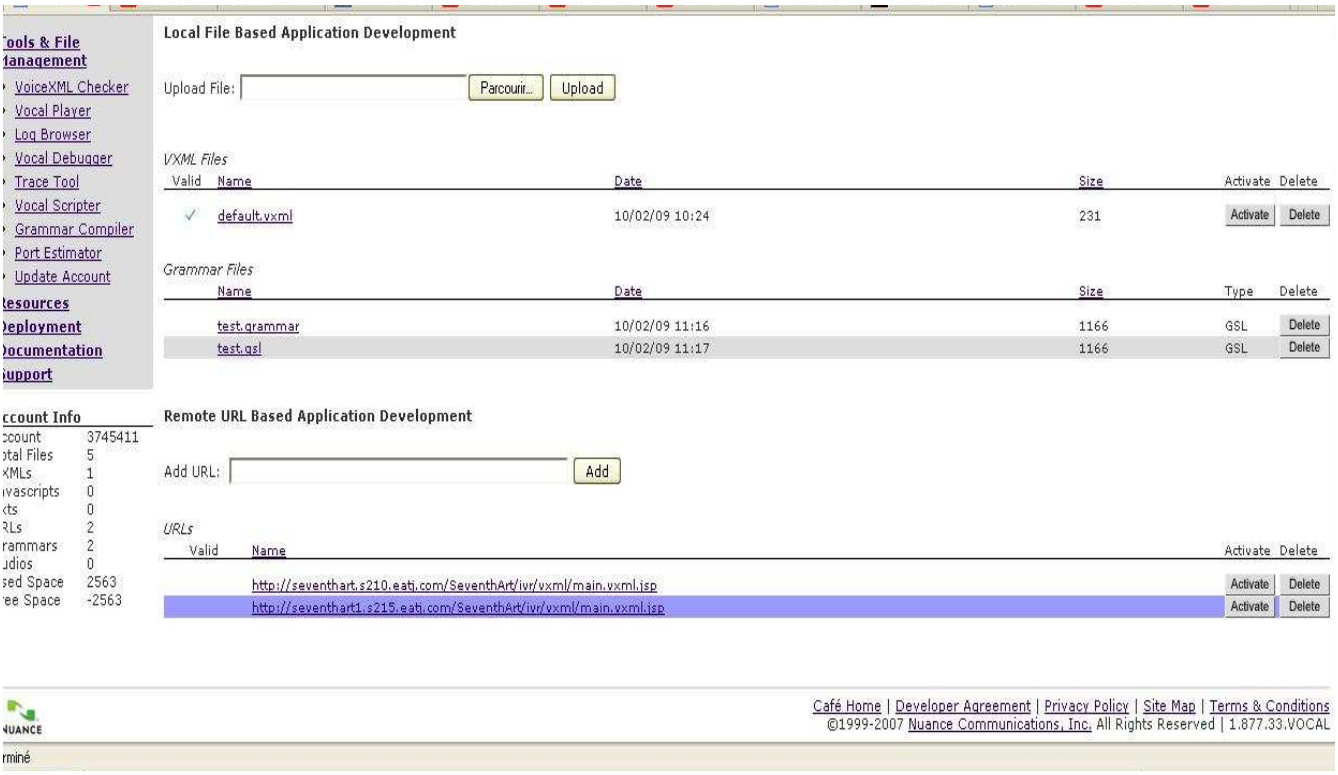


Fig.31: Imagen que muestra la página principal de la plataforma Nuance. La línea en color azul indica el enlace del servidor de ubicación de nuestra aplicación vocal.

Cada llamada que se realiza al sistema, se genera de forma automática un "Log" (denominado así en la página de la plataforma), que tiene un identificador único que le distingue de otros Logs. Dicho de otra forma, cada llamada efectuada tiene asociado un Log único.

En el Log está reflejada toda la traza de ejecución, y podemos ver claramente las interacciones del usuario, las palabras pronunciadas, los errores de implementación, las declaraciones de variables, etc.

Por lo cual, cuando ocurre un error, tras haber realizado una llamada al sistema, accedemos al Log correspondiente, y observamos que está indicado el número de línea donde está el error.

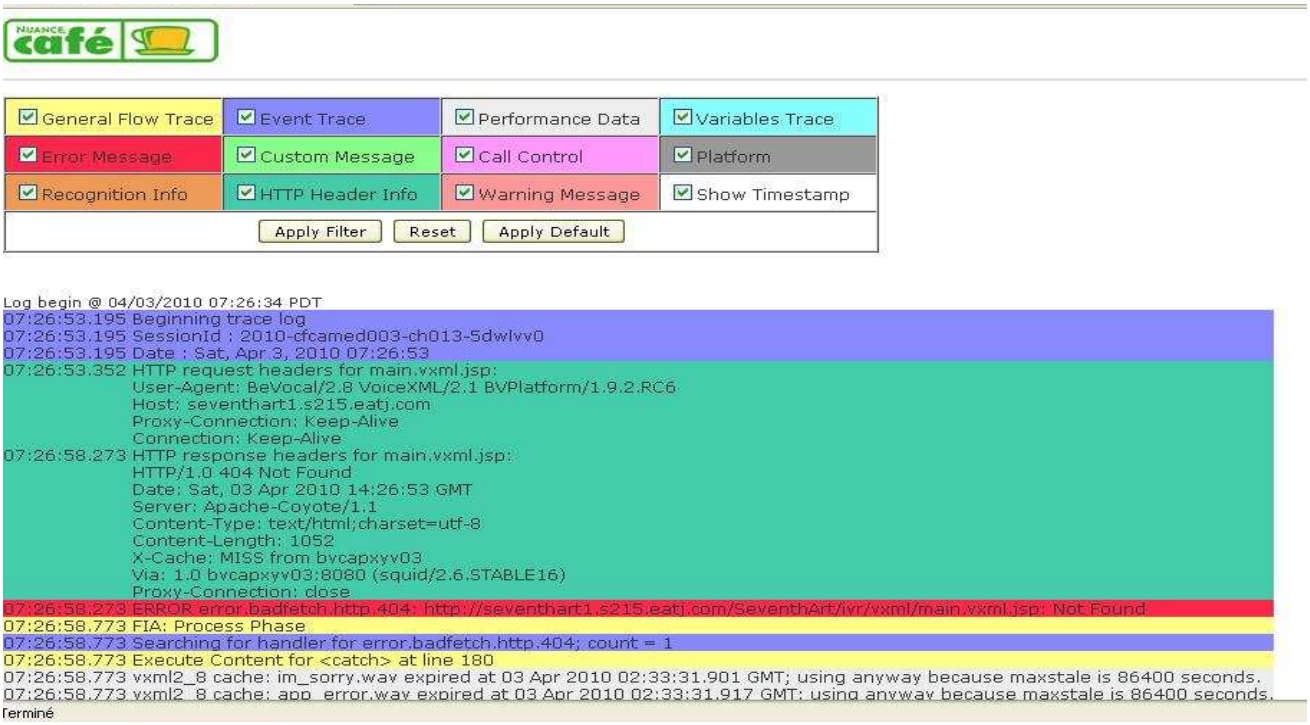


Fig.32: Imagen que indica un Log de una llamada.

Como se puede observar, al principio del Log, podemos encontrar una leyenda que explica el significado de cada color (Ver [Fig.33]).

Además de mostrar las instrucciones ejecutadas durante la llamada, se indica el tiempo de realización de cada una en hora local del Pacífico.

Fig.33: Leyenda asociada a un Log determinado.

Como podemos observar, hay varios colores, sin embargo comentaremos solamente los más importantes que hemos utilizado en nuestras trazas de ejecución.

- ***Error Message :***

  Indica los errores de implementación mostrando el número de línea donde se encuentra éste.

  Esta línea indica que hay un error de implementación de sintaxis JavaScript en la línea 393 del código.

- ***Recognition Info :***

  Representa las frases o palabras pronunciadas por parte del usuario.

  En este ejemplo, se observa claramente que el usuario ha pronunciado la palabra Valencia a la hora de seleccionar la ciudad.

- ***HTTP Header Info:***

  Señala los detalles de cada petición-respuesta del servidor HTTP asociadas a cada fichero utilizado por parte de la aplicación.

```
05:09:59.483  HTTP request headers for welcome.wav:
              User-Agent: BeVocal/2.8 VoiceXML/2.1 BVPlatform/1.9.2.RC6
              Host: seventhart1.s215.eatj.com
              Cookie: $Version=0; JSESSIONID=7557C2990B004A0AB93FB618F6A649D8; $Path=/SeventhArt
              Proxy-Connection: Keep-Alive
              If-Modified-Since: Mon, 15 Mar 2010 20:37:46 GMT
              If-None-Match: W/"127896-1268685466000"
              Connection: Keep-Alive
05:09:59.499  HTTP response headers for welcome.wav:
              HTTP/1.0 304 Not Modified
              Date: Mon, 15 Mar 2010 12:09:59 GMT
              Server: Apache-Coyote/1.1
              ETag: W/"127896-1268685466000"
              X-Cache: MISS from bvcapxyv03
              Via: 1.0 bvcapxyv03:8080 (squid/2.6.STABLE16)
              Proxy-Connection: keep-alive
```

- ***Warning Message:***

  Como su nombre indica, representan les warnings de implementación.

```
05:00:19.448  The MIME Content-Type "text/javascript" is invalid for JavaScript. The proper type is "application/x-javascript".
```

Se trata de un ejemplo de un warning que se ha dado en nuestra aplicación.

- ***Variables Trace:***

  Indica las distintas declaraciones y asignaciones de variables que se han efectuado en el código.

```
05:10:30.858  Assigned context.vars.comingFrom to FilmLookup
05:10:30.858  VAR : Initializing city to expression context.vars.city
```

Según el fragmento mostrado, se puede observar que se ha asignado el valor “FilmLookup” a la variable “context.vars.coming”.

Del mismo modo, se ha inicializado la variable “city” con el valor almacenado en la variable “context.vars.city”.

Conclusión

En este trabajo hemos presentado un sistema que gestiona todos los cines del país mediante una interfaz vocal.

Este sistema ha sido diseñado para facilitar la reserva y la compra de tickets en los distintos cines del país mediante llamadas telefónicas y está dirigido a personas de todas las edades, sin necesidad de tener mucho conocimiento de informática para poder utilizarlo.

Esta aplicación ha sido desarrollada en inglés (los prompts, las gramáticas, etc.). Por lo cual, podemos definir nuestro trabajo futuro que podemos resumir en: desarrollar la versión en español (u otro idioma). Esta mejora necesita: definir las gramáticas en español, grabar los mensajes de audio o prompts en español y definir una interacción (o estado) al principio en la cual el usuario debería elegir un idioma para poder continuar la interacción con el sistema.

Bibliografía

- The Eduteka :

http://www.eduteka.org/modulos.php?catx=4&idSubX=116

- Crossroads The ACM Student Magazine :

http://www.acm.org/crossroads/

- Sonify : Interactice Sound For The Web & Wireless:

http://www.sonify.org

- Página oficial de W3C:

http://www.w3.org/TR/voicexml20/#dmlABuiltins

- http://www.webestilo.com/javascript/

- La Wikipedia :

http://es.wikipedia.org/wiki/JSON

http://es.wikipedia.org/wiki/Voz_sobre_IP

- Página oficial de Bevocal o Nuance :

http://cafe.bevocal.com/

Glosario de términos

- ***ASP:*** **Active Server Pages**, es una tecnología de Microsoft para páginas web generadas dinámicamente.

- ***TTS :*** **Text to Speech**, es una técnica que permite generar oralmente cualquier mensaje en forma de texto mediante un sintetizador de voz.

- ***ASR:*** **Speech Recognition**. Componente de reconocimiento de voz VXML.

- ***DTMF:*** **Dual Tone Multifrecuency**. Tonos en diferentes hertz que utiliza una telefonía para marcar números. Cada número u opción del teléfono tiene su tono que le identificado en la telefonía.

- ***IVR:*** **Interactive Voice Response**, sistema de navegación telefónica interactiva.

- ***VoIP:*** **voice over IP**. Es un grupo de recursos que hacen posible que la señal de voz viaje a través de Internet empleando un protocolo IP (Protocolo de Internet). Esto significa que se envía la señal de voz en forma digital, en paquetes, en lugar de enviarla en forma digital o analógica, a través de circuitos utilizables sólo para telefonía.

UPV
Camino de Vera, s/n
46022 – Valencia
ESPAÑA