Escola d’esquí

Aitor Miranda San Andres
ETIG

Javier Ferró Garcia

Memòria
18/06/2007

Índex

Introducció

Objectius

L'objectiu de la pràctica consisteix en crear una petita aplicació, que en aquest cas consisteix en una aplicació per a la gestió d'una escola d'esquí, utilitzant la tecnologia J2EE.

L'objectiu d'aquest document consisteix en recollir tots els passos necessaris per a poder desenvolupar l'aplicació, des de la pressa de requeriments feta a un hipotètic usuari, fins a alguns detals del desenvolupament de la mateixa.

També es detallarà la planificació de tota la feina necessaria per a poder dur a terme aquesta pràctica, i els diferents pasos de disseny realitzats.

Els meus objectius sobre aquesta pràctica es basen en conèixer l'arquitectura utilizada amb el nom de J2EE per tal de poder crear aplicacions des de zero i aplicar alguns dels patrons de disseny que existeixen actualment.

Requeriments

Comencem per definir els requeriments que ha de complir l'aplicació que volem crear.

L'aplicació ha de permetrer realitzar la gestió interna d'una escola d'esquí.

Aquesta gestió consisteix que a l'inici de temporada es donen d'alta els professors i les persones encarregades de la secretaría que hi treballaran, llavors la secretaría parlarà amb els clients per anotar les classes que volen fer i assignar-los a un professor que els donarà les classes.
Per a poder assignar un professor que estigui lliure han de saber la disponibilitat dels professors, tant durant tota la temporada com l'hora a la que entren o surten de treballar cada dia.
També és necessari poder consultar quantes hores de classe ha fet cada professor en un període de temps per tal de poder comunicar-ho al gestor que realitza les factures per pagar als professors.

Amb això veiem que tindrem tres perfils d'accès:

Administrador:
És el perfil que gestiona tot el control de perfils. Pot donar d'alta, de baixa o modificar els professors i les persones de la secretaría, poden ser possible que un professor pugui estar a la secretaría en un moment donat.
També ha de poder consultar la disponibilitat de tota la gent al llarg de la temporada per saber si algun dia faran curt de professors o de secretaris.

Secretaría:
Gestiona la assignació dels clients als professors, pel que ha de poder veure la disponibilitat dels professors aquell dia, es a dir, els que tenen marcat que aniran a treballar i si ja han arrivat a l'escola. També ha de poder veure l'especialitat i la titulació dels professors per tal d'assignar el més adient al nivell solicitat pel client.
També ha de marcar la seva disponibilitat al llarg de la temporada.

Professor:
És la persona que imparteix les classes d'esquí als clients. Pel que ha de marcar la seva disponibilitat al llarg de la temporada, i també ha de marcar quan arriva i quan s'envà de l'escola per tal que des de la secretaría puguin saber si està disponible per assignar-li alguna classe.
Ha de poder consultar l'horari de classes que tindrà al llarg del dia.
També ha d'anotar-se quina és la seva especialitat, si esquí o snow, i quina titulació té.

Dels clients interesa guardar el nom, un telèfon de contacte (que és opcional), el nivell de la classe solicitada i el número de persones que estaran a la classe (per a grups). També es podrà incloure observacions, com per exemple indicar si és un grup de nens petits.

Les consultes que es poden realitzar han de consistir en un resum d'hores per un període de temps determinat i que es pugui agrupar per: dia, professor, especialitat, nivell del client i quantitat de persones a la classe.
Els criteris opcionals d'aquestes consultes poden ser: professor, especialitat i nivell del client.

Fases i tasques

Un cop tenim els requeriments que ha de complir l'aplicació, dividim la feina que hem de fer per crear l'aplicació en fases, que ja estan fixades, i cadascuna d'aquestes fases es divideix en diferents tasques. Les fases i les seves tasques són:

Anàlisi i disseny:

- Definició dels casos d'ús. On s'especifica la interacció dels usuaris amb el sistema
- Definició del diagrama Entitat-Relació. On tindrem el disseny que ha de tenir la base de dades per guardar totes les dades.
- Definició de la interficíe gràfica. On es farà un esquema estàtic que plasmi la funcionalitat de la aplicació.
- Definició del diagrama de classes. On s'especifica l'estructura de classes necessària per poder després implementar la aplicació.

Implementació:

- Tria de l'arquitectura a utilitzar en l'aplicació.
- Preparació de l'entorn de treball en es desenvoluparà l'aplicació.
- Creació de la base de dades. On es crea la base de dades dissenyada.
- Implementació de la aplicació. On es desenvolupa la aplicació.
- Proves. S'ha de provar que la aplicació es comporta tal i com ha estat definit

Memòria i presentació:

- Creació de la memòria del projecte.
- Creació de la presentació.

Planificació

Ara que tenim les tasques definides hem de planificar quan farem cada tasca per tal de complir les dates clau del projecte.

Les dates de finalització de les fases ja estan fixades. La data final de la fase d'anàlisi i disseny és el 16 d'Abril, la data de la fase d'implementació és el 21 de Maig i la data final és el 18 de Juny.

Partint d'aquestes dates podem distribuir les tasques de la següent manera:

Anàlisi i disseny:

- Definició dels casos d'ús. 19 / 03 / 2007
- Definició del diagrama Entitat-Relació. 26 / 03 / 2007
- Definició del diagrama de classes. 02 / 04 / 2007
- Definició de la interficíe gràfica. 09 / 04 / 2007

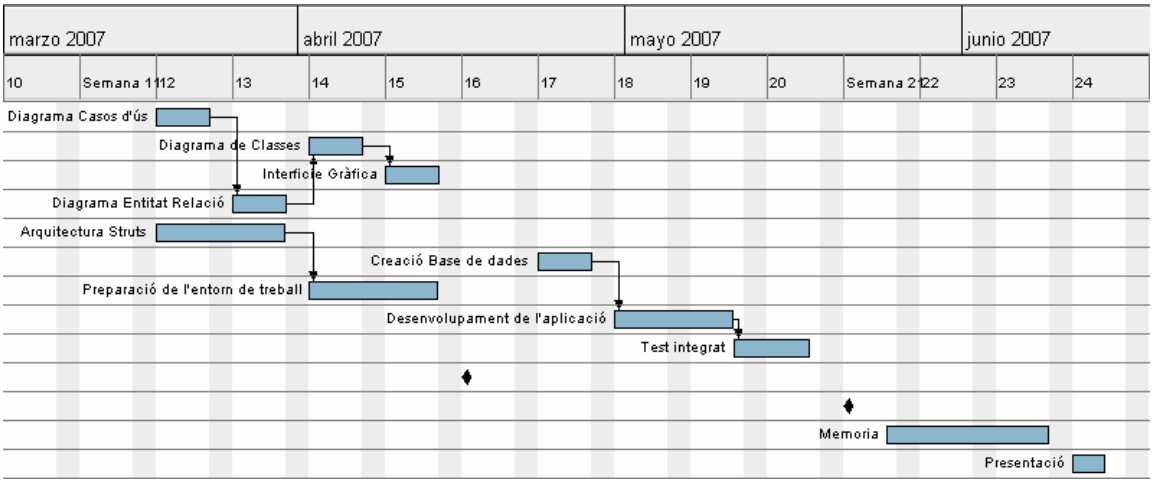
Implementació:

- Creació de la base de dades. 23 / 04 / 2007
- Implementació de la aplicació. 07 / 05 / 2007
- Proves. 14 / 05 / 2007

Memòria i presentació:

- Creació de la memòria del projecte. 04 / 06 / 2007
- Creació de la presentació. 11 / 06 / 2007

A continuació es pot veure la planificació en un diagrama de Gantt amb la dependència entre les tasques i les fites importants de la PAC 2 i la PAC 3.



També hi ha un parell de tasques que corresponen a la part d'implementació que es realitza mentre es fa el disseny ja que corresponent a la tria de l'arquitectura i la preparació de l'entorn de treball.

Arquitectura

Abans de començar el disseny de l'aplicació hem de triar sobre quina arquitectura ho farem.

La implementació d'aquesta aplicació la separarem en tres capes: visual, gestió i dades.

Aquesta separació de l'arquitectura ens permetrà un manteniment de l'aplicació, així com poder canviar alguna de les parts sense tenir que modificar la resta de capes, sempre que se segueixi el mateix criteri.

Per a separar la capa visual de la capa de gestió utilitzaré l'arquitectura Struts 2.
Struts 2 és un framework que ha estat dissenyat per a ajudar al desenvolupament, posta en producció i manteniment d'aplicacions.

A més, Struts 2 utilitza el patró de disseny MVC-2, el qual separa la part visual de la part de gestió, el que és una gran ajuda a l'hora de desenvolupar i mantenir l'aplicació.

Per a separar la capa de gestió de la capa de dades utilitzo el patró de disseny DAO, el que crea una interfície de comunicació entre totes dues capes per tal que siguin independents els detalls de la implementació de cadascuna de les capes.

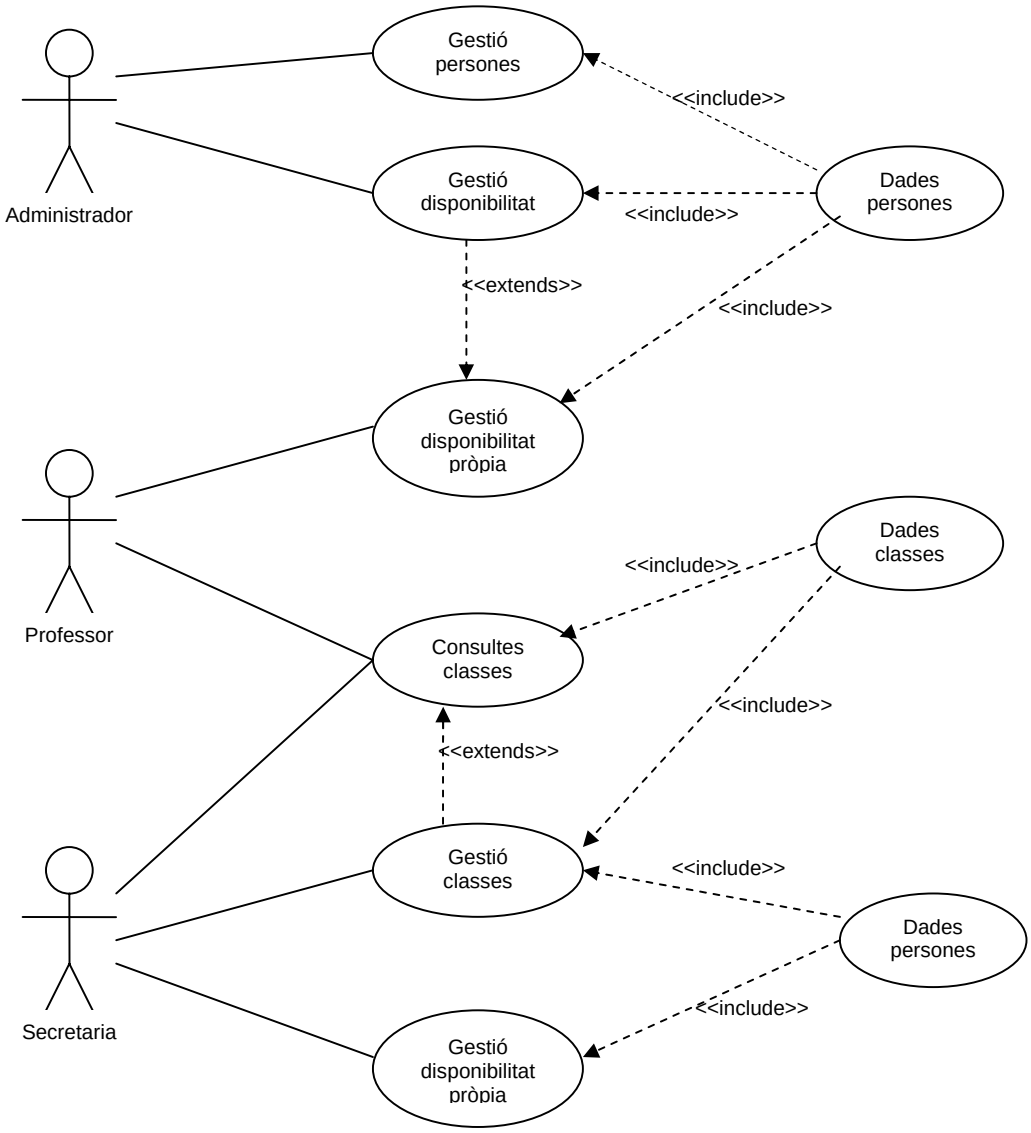
Per a això he creat una aplicació anomenada EsquiDAO que conté la interficie de comunicació i la implementació de la capa de dades, aquesta aplicació está empaquetada en un jar que s'ha d'incloure a la llibreria de l'aplicació aquí desenvolupada, al igual que el jar necessari per a que es pugui conectar a la base de dades, que en aquest cas és un Oracle.

Casos d’ús

Un cop hem triat quina arquitectura aplicarem ja podem començar el disseny.

Els casos d’ús ens defineixen l’interacció que tindran els usuaris amb l’aplicació, amb el que podrem saber què és el que hauria de tenir l’interficie gràfica.

Primer hi ha el diagrama de casos d’ús i a continuació hi ha una descripció dels casos més importants. Hi ha alguns casos d’ús repetits al diagrama per facilitar la visibilitat del mateix, de manera que es pugui entendre amb un cop de vista.



Abans que qualsevol actor pugui realitzar alguna acció s'ha d'haver validat al sistema per tal de comprobar quin és el seu perfil.

Cas d'ús 1: Gestió persones

Actors: Administrador

Consisteix en tota la gestió de persones que poden utilitzar l'aplicació, pel que s'ha de permetre donar d'alta, donar de baixa,consultar i modificar totes les dades d'una persona.

Les dades que ha de tenir tota persona són: usuari (l'usuari ha de ser únic), password, perfil usuari (administrador, professor, secretaria), nom, cognom, NIF, teléfon de contacte, especialitat (esquí o snow) i nivell (asociat, nivell 1, nivell 2 o nivell 3).
Les dades d'especialitat i nivell només són obligatòries per a perfil profesor.

Cas d'ús 2: Gestió disponibilitat

Actors: Administrador, Professor i Secretaria

Aquí es podrá gestionar la disponibilitat de la gent per saber quins dies estaran treballant, pel que s'ha de poder donar d'alta, de baixa, consultar i modificar els dies que cada persona estarà disponible per treballar.

Quan l'actor no és l'administrador l'usuari només podrà veure i modificar les pròpies dades, que correspon al cas d'ús "Gestió disponibilitat pròpia".

Cas d'ús 3: Gestió classes

Actors: Secretaria

Gestió de les classes que s'imparteixen a l'escola d'esquí, amb el que es pot donar d'alta, de baixa, consultar i modificar les classes.
Només es podran modificar totes aquelles classes que encara no s'han impartit.

Per poder donar d'alta una classe hem de tenir algunes dades del client a qui anirà dirigida, com un nom, un telèfon de contacte (opcional), quantes persones formaran part del grup, el nivell de la classe solicitada i l'opció d'afegir comentaris.

I les dades de la classe consisteixen en el professor que impartirà la classe, el client a qui va dirigida, el dia i l'hora de l'inici de la classe, la durada (que serà en hores) i un comentari per si es demana que s'imparteixi la mateixa classe durant més d'un dia.

Cas d'ús 4: Consulta classes

Actors: Secretaria i Professor

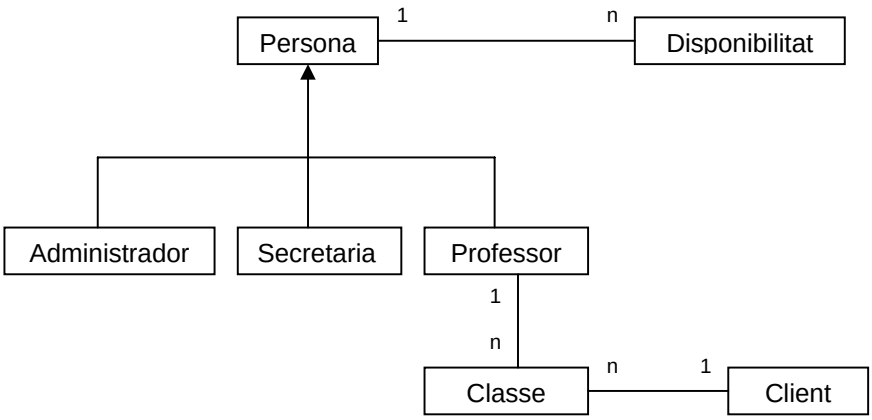
Aquí és on es pot veure quin horari té un professor per a un determinat dia.
També es permet realitzar consultes per saber la quantitat d'hores en un període de temps determinat agrupades per: dia, professor, especialitat, nivell del client o quantitat de persones a la classe.
Els criteris opcionals d'aquestes consultes poden ser: professor, especialitat i nivell del client.

Diagrama Entitat-Relació

Per a poder realitzar el diagrama entitat-relació primer hem d'identificar els objectes de l'aplicació.

Llegint els requeriments podem veure que a l'aplicació es conectarà un usuari que pot tenir tres perfils diferents (administrador, secretaria o professor), també tenim els clients que demanen el servei i les classes que se'ls imparteixen. Finalment veiem que es parla de la disponibilitat dels treballadors per tal de saber quins dies treballen i quin horari fan.
Amb aquesta informació i la relació que tenen entre si podem realitzar el diagrama.

Aquí es pot veure el diagrama entitat-relació de l'estructura que ha de tenir la base de dades.
Després del diagrama es detalla els atributs que ha de tenir cada entitat amb el seu tipus i la seva propietat.



A continuació hi ha els atributs per a cada entitat on la clau primària estarà subrallada, entre parèntesis hi haurà el tipus i si és clau alternativa (AK) o forànea (FK), i en cursiva estaran els atributs que poden pendre el valor null.

Persona: usuari (String), password (String), nom (String), cognoms (String), NIF (String), telefon (String)

Profesor: especialitat (String), nivellProfessor (String)

Classe: profesor (String, FK1 Professor), data (Date), hora (Hora), client (int, FK2 Client), durada (int), *comentari* (String)

Client: id (int), nom (String), *telefon* (String), nombreClients (int), nivellClasse (String), *comentari* (String)

Disponibilitat: persona (String, FK1 Persona), data (Data), *horaEntrada* (Hora), *horaSortida* (Hora)

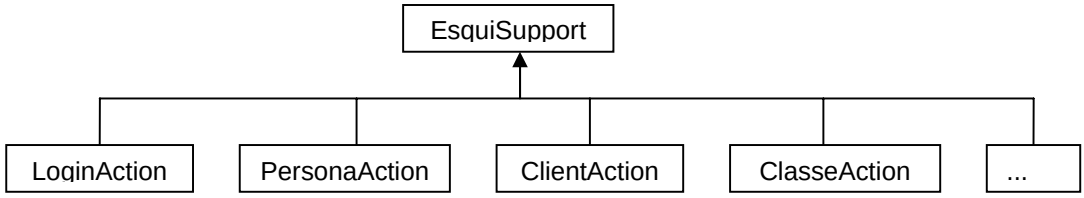
Les classes Administrador i Secretaria no tenen cap atribut a més dels que hereten de Persona.

L'herència la solucionarem amb la creació d'una única taula per a tota la jerarquia ja que així millorarem el rendiment i la pèrdua d'espai és petita degut a que Professor és la subclasse que més persones tindrà.

Diagrama de classes

A continuació es detalla quina estructura de classes tindrà l'aplicació.

Per una banda tenim les classes que són les accions que es criden desde les pàgines jsp amb que l'usuari interactua, que totes tindran la mateixa classe com a base per tal de ajuntar en una única tota la funcionalitat comuna, i llavors, d'aquesta classe base derivaran la resta de classes d'acció. Aquesta estructura de classes ve donada per la utilització d'Struts a l'aplicació.
A continuació es pot veure en un diagrama.



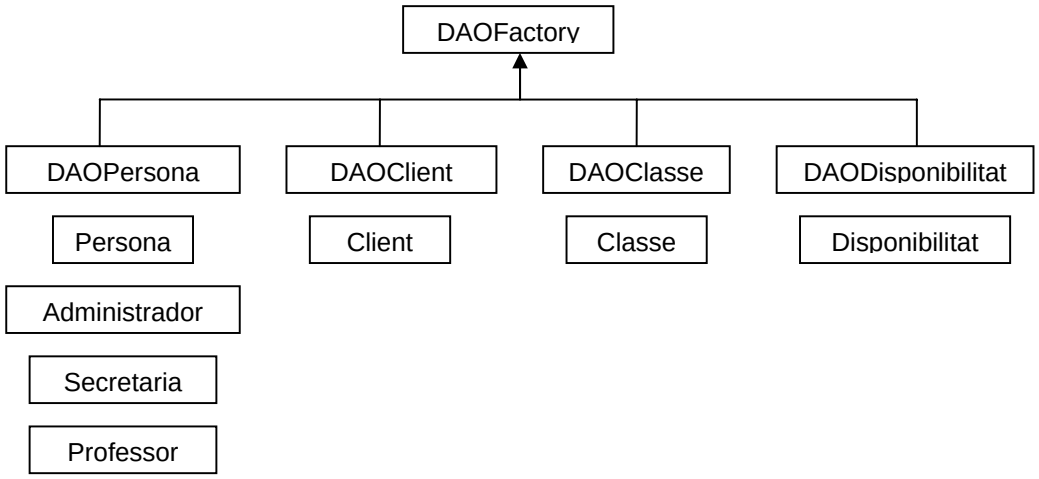
La classe EsquiSupport deriva de ActionSupport, classe que conté cert comportament de struts necessari, com el tractament d'errors. Aquesta classe també implementa les interfícies SessionAware i ApplicationAware per tal de mantenir certa informació sobre l'aplicació i sobre la sessió.

Llavors cada subclasse tindrà els mètodes propis per a cada acció.

De d'aquestes classes s'utilitzarà una altra classe on tenir els paràmetres d'accés a les bases de dades i amb uns mètodes de manera que realitzin les accions contra la base de dades i retornin els objectes definits al diagrama d'entitat-relació.

Ho faré de manera que separaré cada entitat amb una classe per tal que sigui més fàcil canviar la base de dades utilitzades sense canviar l'aplicació. I per a tal fi també faré que l'aplicació només conegui unes interfases i les entitats, amb el que encara queda més separat el que és la part del negoci de l'aplicació de la part de dades.

A continuació hi ha un diagrama de classes on es pot veure la relació entre les classes d'accés a la base de dades, on també es pot veure a sota de cadascuna d'elles les entitas que tractaran.



Interfície gràfica

Un cop hem dissenyat com es farà l'aplicació hem de definir la interfície amb l'usuari.

La interfície gràfica constarà de pàgines web, pel que s'accedirà a l'aplicació mitjantçant un navegador d'internet.

Ara farem la interfície gràfica senzilla i funcional deixant per més endavant l'ampliació d'aquesta part, ja que per fer-ho requereix coneixements amplis de JSP, HTML i Javascript, cosa que queda fora de l'àmbit d'aquest projecte.

Al conectar-se a l'aplicació aparaixerà una pàgina donant la benviguda a l'aplicació i on es podrà introduïr l'usuari i password per poder-se conectar.

Un cop l'usuari ja estigui conectat anirà a una pàgina on estarà el menu de totes les accions que pot fer, i des d'on pasarà a la resta de pàgines.

Si va a la gestió de persones, veurà una llista de totes les persones que estan donades d'alta a l'aplicació i des d'on podrà fer el manteniment, alta, baixa i modificació, un cop triada quina d'aquestes accions vol fer pasarà a una altra pantalla on veurà totes les dades de la persona triada (si no fa una alta) i podrà realitzar l'acció o cancelar tornant a la pàgina anterior.
Si l'usuari no és l'administrador, quan entri a la gestió de persones pasarà directament a la pantalla de modificació per tal que pugui canviar les pròpies dades personals.

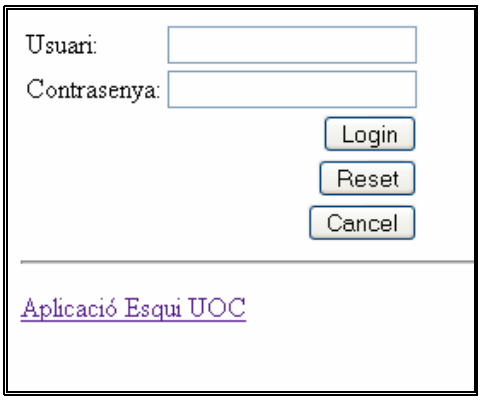
Si tria la gestió de la disponibilitat pròpia veurà una llista de les dates en les que diu estar disponible (posterior o igual al dia d'accés), i des d'on podrà donar d'alta o de baixa altres dates.
Si l'usuari és un professor podrà marcar l'hora d'entrada del propi dia, si encara no ho ha fet, i marcar l'hora de sortida si ja ha marcat l'hora d'entrada.
Si l'usuari és l'administrador, abans de veure la llista de dates disponibles, tindrà una llista de les persones per tal que trii sobre qui vol realitzar la gestió de disponibilitat.

Si es tria la gestió de classes sortirà una llista amb les classes que comencen a l'hora actual o posterior per tal de poder modificar, donar d'alta o de baixa les classes, amb el que aniria a una altra pàgina on es veurien les dades de la classe i del client que l'ha solicitat.

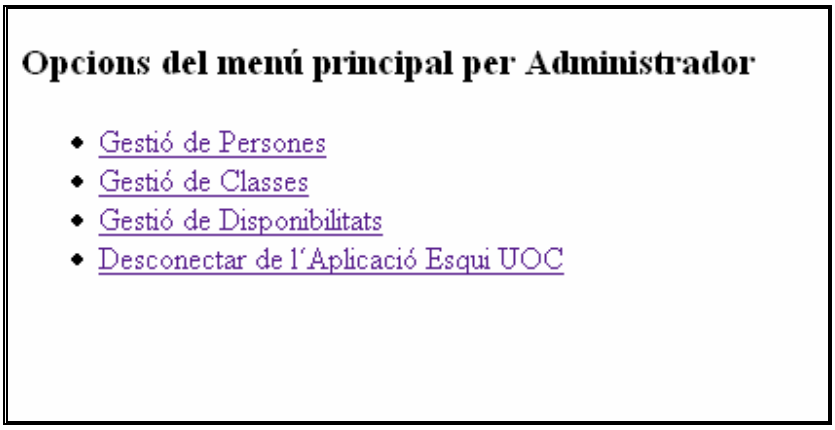
Des de la gestió de classes o des de el menu principal es podrà accedir a la consulta de classes on primer hi haurà una pàgina amb els criteris i tipus de consulta possibles, i un cop triats es realitzarà la consulta passant a una segona pàgina on es veuran els resultats.

A continuació es poden veure algunes de les pantalles de l'aplicació:

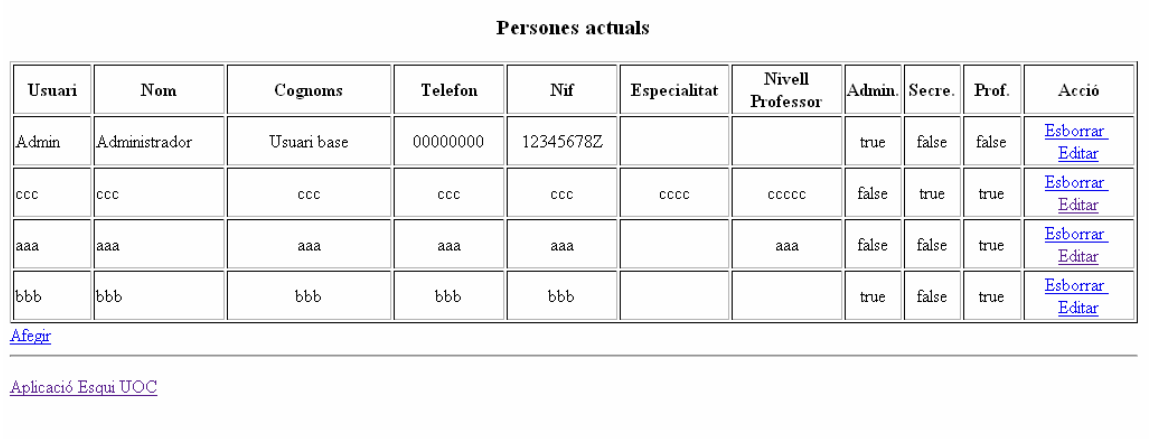
Pàgina de login:



Menú principal (per a perfil administrador):



Llista de persones de la opció de Gestió de Persones:



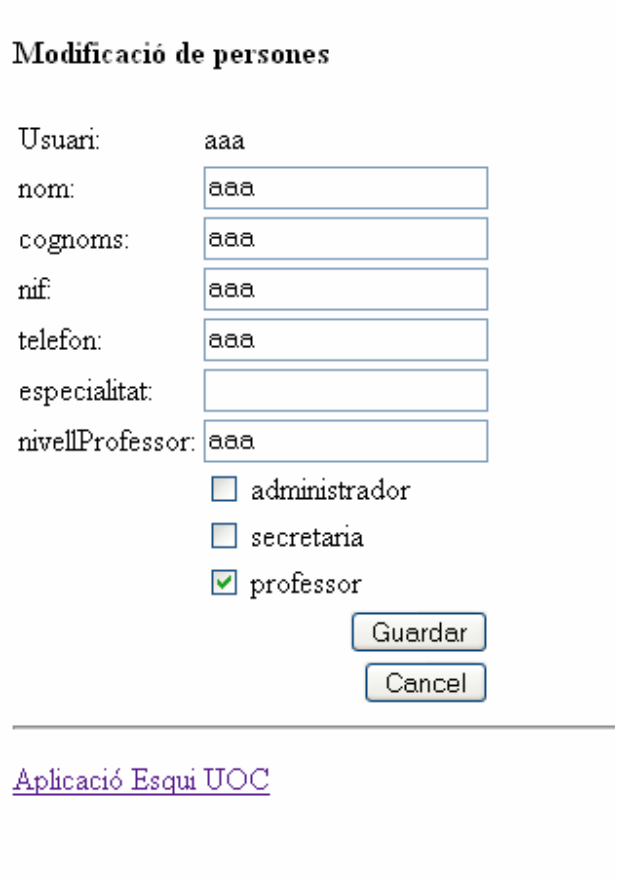
Persones actuals

Usuari	Nom	Cognoms	Telefon	Nif	Especialitat	Nivell Professor	Admin.	Secre.	Prof.	Acció
Admin	Administrador	Usuari base	00000000	12345678Z			true	false	false	Esborrar Editar
ccc	ccc	ccc	ccc	ccc	cccc	ccccc	false	true	true	Esborrar Editar
aaa	aaa	aaa	aaa	aaa		aaa	false	false	true	Esborrar Editar
bbb	bbb	bbb	bbb	bbb			true	false	true	Esborrar Editar

Afegir

Aplicació Esqui UOC

Menú per a la modificació de dades d'una Persona:



Les altres pàgines de l'aplicació segueixen el mateix criteri que les aquí mostrades, canviant únicament les dades que es mostren.

Creació de la base de dades

El primer pas per al desenvolupament de l'aplicació consisteix en crear la base de dades on guardarem la informació.

Un cop tenim el diagrama d'entitat-relació ja podem procedir a implementar la base de dades utilitzant un gestor de bases de dades.

Per a aquesta aplicació he utilitzat un Oracle, peró es podria utilitzar qualsevol altre canviant la implementació de la capa de dades per tal que utilitzi el nou gestor de base de dades triat.

Per a la implementació de la base de dades crearem dos usuaris, un que serà l'administrador d'aquesta i l'altre que serà l'usuari utilitzat per l'aplicació per connectar-s'hi i poder-hi interactuar.

Per a tal fi he creat un tablespace anomenat UOC_TFC, amb l'usuari root de l'Oracle, en el que crearem la base dades amb la que treballarem. I amb el mateix usuari root creem els dos usuaris indicats i els hi donem els permisos que necessiten:

```
--Creació de l'usuari amb el que administrarem la base de dades d'esqui
CREATE USER "ESQUI_ADMIN" IDENTIFIED BY "ESQUI_ADMIN"
 DEFAULT TABLESPACE UOC_TFC
 QUOTA  UNLIMITED  ON UOC_TFC;

GRANT CREATE SESSION TO "ESQUI_ADMIN";
GRANT CREATE TABLE TO "ESQUI_ADMIN";
GRANT CREATE SEQUENCE TO "ESQUI_ADMIN";


--Llavors creem l'usuari per a l'aplicació, amb el
--que farem les consultes i les modificacions de les dades
CREATE USER "ESQUI_USER" IDENTIFIED BY "ESQUI_USER"
 DEFAULT TABLESPACE UOC_TFC
 QUOTA  UNLIMITED  ON UOC_TFC;

GRANT CREATE SESSION TO "ESQUI_USER";
```

Un cop tenim els dos usuaris creats ens conectem amb l'usuari administrador per crear les taules i donar permisos d'accés a aquestes taules a l'usuari de l'aplicació:

```
CREATE TABLE ESQUI_ADMIN.PERSONA
(
	usuari  VARCHAR2(16) NOT NULL,
	password	VARCHAR2(16) NOT NULL,
	nom	VARCHAR2(30) NOT NULL,
	cognoms VARCHAR2(60) NOT NULL,
	NIF	VARCHAR2(10) NOT NULL,
	telefon VARCHAR2(15) NOT NULL,
	tipus	INT NOT NULL,
	especialitat	VARCHAR2(10),
	nivellProfessor	VARCHAR2(15),
	CONSTRAINT PK_PERSONA PRIMARY KEY (usuari )
)
;

CREATE TABLE ESQUI_ADMIN.CLASSE
(
	professor	VARCHAR2(16) NOT NULL,
	data	DATE NOT NULL,
	client  INT NOT NULL,
	durada  INT NOT NULL,
	comentaris	VARCHAR2(250),
	CONSTRAINT PK_CLASSe PRIMARY KEY (professor, data )
)
```

```
;

CREATE TABLE ESQUI_ADMIN.CLIENT
(
       id     INT NOT NULL,
       nom    VARCHAR2(60) NOT NULL,
       telefon VARCHAR2(15),
       nombreClients  INT NOT NULL,
       nivellClasse   VARCHAR2(15) NOT NULL,
       comentaris     VARCHAR2(250),
       CONSTRAINT PK_CLIENT PRIMARY KEY (id )
)
;

CREATE TABLE ESQUI_ADMIN.DISPONIBILITAT
(
       persona VARCHAR2(16) NOT NULL,
       data   DATE NOT NULL,
       horaEntrada    DATE,
       horaSortida    DATE,
       CONSTRAINT PK_DISPONIBILITAT PRIMARY KEY (persona, data )
)
;

--Creem el codi seqüència per a l'identificador de clients
CREATE SEQUENCE S_CLIENT
       INCREMENT BY 1
       START WITH 1;


--I donem permisos a les taules a l'usuari de l'aplicació
GRANT SELECT, INSERT, UPDATE, DELETE ON CLIENT TO esqui_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON CLASSE TO esqui_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON PERSONA TO esqui_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON DISPONIBILITAT TO esqui_user;
GRANT SELECT ON S_CLIENT TO esqui_user;

--Finalment Insertem un usuari com a administrador per tal que es
--pugui utilitzar l'aplicació
INSERT INTO PERSONA ( USUARI, PASSWORD, NOM, COGNOMS, NIF, TELEFON, TIPUS, ESPECIALITAT,
NIVELLPROFESSOR ) VALUES (
'Admin', 'Admin', 'Administrador', 'Usuari base', '00000000T', '00000000', 1, NULL
, NULL);
COMMIT;
```

Amb això ja tenim la base de dades creada per a poder-la utilitzar en l'aplicació.

Ara cal crear un fitxer de propietats per a indicar com es pot conectar a la base de dades.

Aquest fitxer de propietats l'he anomenat "database.properties" i ha de contenir les següents dades:

- host Nom de la màquina on es troba l'oracle
- port Port de conexió amb l'oracle
- server Nom del servidor de l'oracle on es troba la base de dades
- user Nom de l'usuari de l'aplicació ("ESQUI_USER")
- passwd Contrasenya de l'usuari de l'aplicació ("ESQUI_USER ")

Implementació de la aplicació

Ara que ja tenim la base de dades creada, ja podem prodecir a implementar la solució triada.

La implementació de la aplicació es divideix en dos parts, la implementació de la capa de negoci seguint Struts i la implementació de la capa de dades seguint el patró de disseny DAO.

Implementació capa de dades:

A la capa de dades tenim les classes que representen els objectes que s'han d'emmagatzemar a la base de dades, i que representen a les taules que s'ha creat.
Aquests objectes són els que es comuniquen entre la capa de dades i la capa de negoci.

També tenim una classe abstracta "DAOFactory" que és des d'on obtenim uns objectes amb els que demanarem l'accés a les dades.
La capa de negoci només coneix la classe abstracta i la interfície dels objectes creats per aquesta, d'aquesta manera es pot canviar la base de dades sense que la capa de negoci se'n vegi afectada ja que només tindriem que canviar la implementació de la recuperació de les dades mantenint la mateixa interficie.

Per obtenir una instància que representa a la classe abstracta "DAOFactory" se li ha de passar el fitxer de configuració per tal que sàpiga com es pot conectar a la base de dades.

A partir d'aquesta instància podem obtenir els objectes per a accedir a les taules DB2, que les he separat per a cada taula. Amb cadascun d'aquest objectes es pot realitzar una consulta per clau i una inserció, modificació o esborrat d'un objecte, a més d'algunes consultes que retornin més d'un registre.

Implementació capa de negoci:

Com a conseqüència d'haver utilitzat struts per a realitzar la capa de negoci de l'aplicació, hem tingut que definir una classe que implemente la interficie "ServletContextListener" per tal que al arrencar l'aplicació crei una instancia de la base de dades, que en aquest cas és una instancia obtinguda de la classe "DAOFactory". Aquesta classe t'ambé s'encarrega de passar el fitxer de propietats de la base de dades a la creació de la instància d'aquesta, com d'eliminar la referencia a la base de dades un cop es tanca l'aplicació.

També hem definit una altra classe que implementa la interficie "Interceptor" per que s'encarregui de comprobar que s'ha conectat un usuari cada vegada que que es carrega una pàgina, d'aquesta manera no permetrem que puguin realitzar cap acció accedint a les pantalles posteriors a la de login.

La classe "EsquiSupport" és la base per a la resta de classes perquè és aquí on s'implementen les accions comunes per a la resta de pantalles com l'acces a la base de dades, l'acces a les propietats de la sessió y l'ús de propietats comunes a varies pantalles.

Les propietats de la sessió es guarden en un Map que es manté d'una pantalla a una altra, sempre que es mantingui la connexió amb l'aplicació.

En aquesta classe s'ha definit una propietat anomenada "Task" per a poder diferenciar quina acció s'ha de realitzar a una pantalla determinada, i així saber si estem creant, modificant o esborrant

una dada determinada. També s'utilitza per a saber quin perfil té l'usuari connectat i així mostrar nomes les accions que té disponibles.

Per a la gestió de les dades de les Persones, Disponibilitats i Classes tenim una única classe per a cada tipus de dada, tot i que es criden des de dos tipus de pantalles diferents.
Tenim una pantalla en la que es mostren les dades disponibles per a modificar i una altra pantalla en la que es poden modificar o introduïr dades noves.

Implantació de l'aplicació

Un cop tenim tota l'aplicació desenvolupada s'han d'explicar els passos necessaris per tal de poder-la utilitzar.

I abans de poder utilitzar l'aplicació ens hem d'asegurar que tenim tot el necessari per a que pugui funcionar. Pel que necesitarem:

- Un servidor de bases de dades Oracle, que és sobre el que s'ha fet l'accés a dades.
- Un servidor d'aplicacions, que jo he utilitzat un anomenat "Jetty".

A la carpeta amb tots els fitxer de l'aplicació hi ha un document word on es detalla els passos necessaris per a poder utilitzar aquesta aplicació.

A dins de la carpeta "esqui" que acompanya aquest document trobarem diferents carpetes i documents que detallo a continuació:

- carpeta "EsquiDAO" : conté el codi font i el compilat de l'aplicació d'accés a la base de dades.
- Carpeta "src" : conté el codi font d'aquesta aplicació.
- Carpeta "sql" : conté els scripts per a la creació de la base de dades.
- Fitxer "Manual Instalació.doc" : Conté les instruccions per a la instalació i configuració de l'aplicació.
- Carpeta "pages" : conté els fitxers .jsp de la interfície gràfica.
- Carpeta "WEB-INF" : conté l'executable i les llibreries necessàries per a l'aplicació.
- Fitxer "index.html" : és la pàgina d'inici de l'aplicació.

Les carpetes "pages" i "WEB-INF", i el fitxer "index.html" s'utilitzen per a fer el deply de l'aplicació.

Conclusions

Amb aquest projecte he vist la base de l'arquitectura J2EE utilitzant el framework de Struts amb el que he aprés a fer petites aplicacions seguints aquest model.

He vist que l'arquitectura J2EE es basa en pàgines JSP que es veuen en un navegador d'internet i que a través del fitxer de configuració struts.xml es relacionen amb classes que s'encarreguen de realitzar les accions pertinents.

També he aprés que per tal de poder utilitzar aquestes aplicacions fa falta un servidor d'aplicacions, com el Jetty, on es fa el deploy per tal que sigui accessible al navegador.

I he vist el patró de disseny DAO per tal d'encapsular l'acces a les dades per tal de separar la gestió de la base de dades i facilitar el posterior manteniment.

Amb el que he aprés les nocions bàsiques per a poder desenvolupar altres aplicacions amb les que aprofundir en el coneixement de l'arquitectura J2EE.

Bibliografia

Struts – 2 : http://struts.apache.org/2.x/

Eclipse : http://www.eclipse.org/

GanttProject : http://ganttproject.biz/

Oracle : http://www.oracle.com/pls/db92/db92.homepage

Java : http://java.sun.com/

Jetty : http://www.mortbay.org/