Programari de consulta *offline* al campus virtual de la UOC.

Antoni Pifarré Mata
Grau d'Enginyeria Informàtica

Xavier Aracil Díaz
David Gañan Jimenez

13 de gener del 2014


Aquesta obra està subjecta a una llicència de Reconeixement-NoComercial-SenseObraDerivada 3.0 Espanya de Creative Commons

FITXA DEL TREBALL FINAL

Títol del treball:	*Programari de consulta offline al campus virtual de la UOC*
Nom de l'autor:	*Antoni Pifarré Mata*
Nom del consultor:	*Xavier Aracil Díaz, David Gañan Jimemez*
Data de lliurament (mm/aaaa):	*01/2014*
Àrea del Treball Final:	*05.641 Eines del Campus UOC*
Titulació:	*Grau d'Enginyeria Informàtica*

Resum del Treball (màxim 250 paraules):

L'objectiu del projecte és el desenvolupament d'una aplicació d'escriptori que permeti a l'estudiant la consulta *offline* de les darreres novetats disponibles a les aules del campus UOC.

Es tracta d'automatitzar el procés de consulta i descàrrega local de dades per tal de poder consultar-les posteriorment fora de línia.

En general pot ser interessant per qualsevol usuari que estigui interessat en tenir una còpia local dels materials i recursos de comunicació de les diferents aules, però, ho és especialment per aquell perfil d'usuari amb un equip portàtil i que sovint no disposa de massa temps ni de connexió a Internet.

Abstract (in English, 250 words or less):

The goal of the project is to develop a desktop application which allows UOC students to make offline queries about the latest available news within the classrooms of the UOC campus.

It's main purpose is to automate the process of query and to download local data in order to view it later offline.

In general, it can be interesting to everyone who is interested in having a local copy of the content as well as the communication resources of any classroom, but it's especially suited for a users who can lack availability of an Internet connection.

Paraules clau (entre 4 i 8):

Campus, Aules, Recursos, UOC, Offline, Consulta

Índex.

Llista de figures

1. Introducció

1.1 Justificació.

Actualment les aplicacions basades en serveis web són cada cop més freqüents a Internet. Personalment estic convençut que aquest paradigma tindrà, els propers anys, una importància cabdal en el món de la programació i esdevindrà un punt d'inflexió en la forma d'entendre, no solsamènt el desenvolupament d'aplicacions, sinó també les noves formes de negoci basades en serveis web especialitzats. Per aquest motiu conèixer aquest serveis i les tecnologies que els envolta podria ser imprescindible per qualsevol enginyer informàtic.

La UOC ofereix la possibilitat de cursar el TFG en una àrea anomenada Eines del campus UOC. Aquestes eines permeten accedir programàticament a diverses funcionalitats del campus mitjançant una API[1]. És doncs, una possibilitat immillorable per aprendre, sobre una plataforma web de producció i totalment funcional, com accedir i utilitzar una API mitjançant els serveis que facilita.

El projecte, objecte d'aquest treball, ha de permetre la consulta fora de línia dels nous recursos pujats al campus. Es tracta d'establir una connexió automàtica amb el campus virtual de la UOC i fer la descàrrega, de forma sincronitzada, d'aquests recursos per després emmagatzemar-los a la nostra computadora de forma local.

Aquesta funcionalitat és interessant per qualsevol usuari del campus que vulgui tenir una còpia local dels materials i recursos de comunicació de les aules, però, ho és especialment per aquell perfil d'usuari amb un equip portàtil i que sovint no disposa de connexió a Internet.

Actualment no hi ha cap aplicació que permeti emmagatzemar localment els recursos de les aules, especialment els de comunicació. Penso que pot ser força interessant poder consultar i salvar els recursos de qualsevol assignatura fins i tot un cop cursada.

[1] ***A**plication **P**rogramming **I**nterface*

1.2 Objectius del Treball

Les funcionalitats del projecte són:

- Accedir, de forma automàtica, al campus virtual i descarregar-ne les darreres novetats localment.

Un cop fet això ha de permetre consultar les novetats *offline*:

- Accés al calendari i a les activitats.
- Accés a la bústia de correu per veure els missatges.
- Accés a les aules. Missatges i materials.

1.3 Enfocament i mètode.

L'objectiu del projecte consisteix en permetre a l'estudiant de la UOC interactuar amb els components de l'aula. Per aconseguir-ho necessitem utilitzar les funcionalitats que ens proporciona l'**Open API** de la UOC. La pàgina web de desenvolupament del campus UOC és:

http://blogs1.uoc.es/developer/

En aquesta web hi ha tota la documentació necessària per utilitzar l'API. A més podem hi podem trobar exemples i suport tècnic.

La primera qüestió a resoldre consisteix en obtenir accés segur a l'API. Per poder aconseguir-ho necessitem demanar una clau d'accés a la següent adreça web:

http://blogs1.uoc.es/developer/documentacio/demana-una-clau-dacces-a-la-api/

L'API està basada en l'arquitectura **REST**[2] pensada per a sistemes distribuïts basats en hipermèdia. L'accés a l'API utilitza el protocol **OAuth**[3] **2.0** que permet l'autorització segura d'una API per aplicacions d'escriptori, mòbils i web. Qualsevol llenguatge que permeti peticions **HTTP** pot ser utilitzat en una aplicació client. Actualment hi ha llibreries disponibles per als llenguatges *C, C#, Java & Objective-C*. També permet el desenvolupament d'aplicacions per a dispositius mòbils *Android* i *iOS*.

Un cop tenim accés a l'API podem accedir a les dades del portal virtual de la UOC. L'aplicació haurà de fer peticions a la web virtual de la UOC mitjançant l'API i posteriorment analitzar i tractar les dades retornades per a poder utilitzar-les en el projecte.

[2] ***R**epresentational **S**tate **T**ransfer.*
[3] ***O**pen **A**uthorization*.

Un cop tinguem les dades, amb el format adequat al llenguatge de programació, podrem presenta-les mitjançant objectes de la llibreria gràfica.

Un llenguatge candidat per al desenvolupament del projecte podria ser **Java.** És un llenguatge orientat a objecte, multiplataforma i àmpliament utilitzat per al desenvolupament d'epilacions gràfiques d'escriptori**.**

1.4 Planificació del Treball

El projecte es durà a terme en el període comprés entre les dates 19/09/13 fins al 13/01/14. Les diferents etapes s'han agrupat tenint en compte la càrrega i els lliuraments de les diferents *PACs*. La descripció detallada de les diferents tasques, així com de la seva durada i les dates d'inici i fi, les podem observar en la següent taula:

TOTAL	**89 días**	**jue 19/09/13**	**lun 13/01/14**
PAC1	**8 días**	**jue 19/09/13**	**lun 30/09/13**
Pla del projecte	8 días	jue 19/09/13	sáb 28/09/13
Requeriments funcional	1 día	sáb 28/09/13	dom 29/09/13
Documentació	1 día	sáb 28/09/13	dom 29/09/13
Lliurament	0 días	lun 30/09/13	lun 30/09/13
PAC2	**21 días**	**mar 01/10/13**	**lun 28/10/13**
Anàlisi	10 días	mar 01/10/13	dom 13/10/13
Disseny	10 días	mar 15/10/13	dom 27/10/13
Prototip	7 días	vie 18/10/13	dom 27/10/13
Documentacio	5 días	mar 22/10/13	dom 27/10/13
Lliurament	0 días	lun 28/10/13	lun 28/10/13
PAC3	**37 días**	**mar 29/10/13**	**lun 16/12/13**
Implementació	30 días	mar 29/10/13	vie 06/12/13
Proves	5 días	sáb 07/12/13	jue 12/12/13
Documentació	2 días	vie 13/12/13	dom 15/12/13
Lliurament	0 días	lun 16/12/13	lun 16/12/13
Entrega final	**20 días**	**mar 17/12/13**	**lun 13/01/14**
Versió final codi	11 días	mar 17/12/13	mar 31/12/13
Memòria	6 días	mié 01/01/14	mié 08/01/14
Presentació	3 días	jue 09/01/14	dom 12/01/14
Lliurament	0 días	lun 13/01/14	lun 13/01/14

Figura 1 – Planificació del treball.

Aquesta planificació segueix el model iteratiu i incremental. Aquest model fa un refinament successiu en cada etapa del projecte, passant altre cop per les etapes d'anàlisi i disseny. L'objectiu d'aquest model és apropar-nos cada cop més al final del projecte sense pèrdues importants de temps per culpa d'errors en etapes anteriors.

Diagrama de Gantt del projecte.

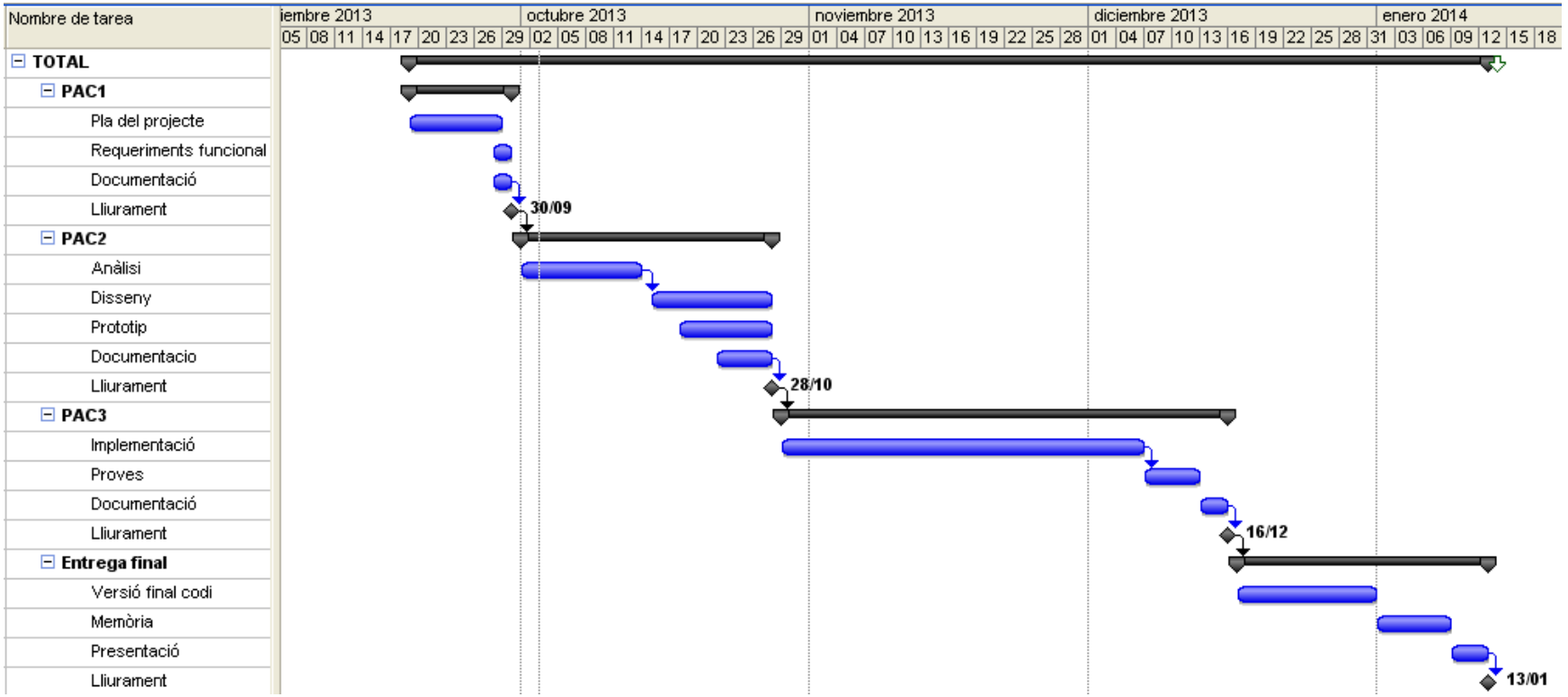

Figura 2 - Diagrama de Gantt.

1.5 Breu sumari de productes obtinguts.

La realització del projecte ha generat diversos productes. Alguns en un format de tipus document digital i altres en un format de tipus programari.

Els productes en format document digital els tenim en aquest mateix document de memòria i són:

- El pla de treball.
- La anàlisi.
- El disseny.
- La implementació del projecte.
- El joc de proves.
- Un apartat amb problemes i dificultats.
- Un apartat amb els canvis respecte el projecte inicial.
- Propostes de millora.
- Conclusions.
- Annexos.

Els productes en format programari són:

- El codi font del projecte, amb el format de *l'IDE Eclipse*.
- La documentació del projecte generada amb *Javadoc*.
- Una carpeta amb les llibreries externes utilitzades (**.jar*).
- Una carpeta amb el codi executable, que inclou:
    - Les instruccions d'instal·lació.
    - L'executable (jar).
    - El programari *xulrunner* per a Windows.
    - Un fitxer *scriptBD.sql* per crear la Base de Dades.

1.6 Breu descripció dels altres capítols de la memòria.

Aquest document s'estructura en nou parts:

- Introducció (objectius, enfocament i planificació).
- Requisits (funcionals i no funcionals).
- Anàlisi (*l'API* de la *UOC*, casos d'ús i el diagrama estàtic d'anàlisi).
- Disseny (Arquitectura, capes, prototipatge, flux de pantalles i el model conceptual de la Base de Dades).
- Implementació (Decisions tecnològiques, la capa de presentació, la capa de domini, la capa de dades, obtenir el *token*, la classe *SwingWorker*, les consultes a *l'API* i les classes genèriques, el *debugger*, el control d'errors i la robustesa, el joc de proves, les dificultats i problemes i les propostes de millora).
- Finalment hi ha els apartats de conclusions, glossari, bibliografia i annexos.

2. Requisits.

2.1 Requisits funcionals.

Un cop definits els objectius podem especificar els requeriments funcionals que caldrà implementar en el projecte:

- Permetre un accés segur, per part de l'usuari, al campus virtual.
- Accedir i emmagatzemar localment en una BBDD les darreres novetats aparegudes al campus.
- Permetre consultar fora de línia tota la informació descarregada localment. Aquestes consultes permetran les següents funcionalitats:
- Lectura dels missatges pendents a la bústia de l'usuari.
- Lectura dels missatges pendents al fòrum i tauler de les assignatures.
- Visualització del calendari per un accés ràpid a les activitats recents.

El programari oferirà aquestes funcionalitats des d'un entorn gràfic que permetrà un accés còmode i segur.

2.2 Requisits no funcionals.

Un cop hem vist els requisits funcionals passem a veure els requisits no funcionals. Aquests requisits són bàsicament:

- Temps de resposta
- No bloqueig de la interfície gràfica.
- Disseny gràfic adaptatiu.
- Robustesa i seguretat.

2.2.1 Temps de resposta.

El temps de resposta quan intervenen xarxes i connexions a Internet pot ser un factor important en la nostra aplicació. Per aquest motiu hem d'aconseguir que les tasques pesades (*Worker Threads*) no esdevinguin un problema en el temps de resposta.

Per aconseguir una bona resposta, en aquest tipus de tasques pesades, s'utilitzarà *threads* d'execució en segon pla.

2.2.2 No bloqueig de la interfície gràfica.

Lligat amb l'apartat anterior cal tenir en compte que Swing utilitza un únic fil d'execució per la part gràfica anomenat ETD (*Even Dispathc Thread*). Qualsevol tasca pesada pot acabar sobrecarregant aquet fil i deixant la interfície gràfica bloquejada. Així doncs, caldrà tenir especial cura i executar les tasques pesades en fils secundaris o *background threads.*

2.2.3 Disseny gràfic adaptatiu.

El disseny de les interfícies gràfiques utilitzaran *Layouts* de posicionament, concretament s'utilitzarà la classe *GridBagLayout*. Aquests gestors posicionen els objectes de forma relativa. Per aquest motiu s'adaptaran a qualsevol resolució i tipus de pantalla.

2.2.4 Robustesa i seguretat.

Per aconseguir una aplicació robusta i segura s'utilitzaran eines de control d'excepcions (*try..catch*, *throws*,...) i d'autenticació segura (OAuth).

3. Anàlisi.

3.1 L'*API* de la UOC.

Per fer l'anàlisi del nostre projecte hem d'entendre els serveis i el model de dades que ens ofereix l'Open API de la UOC. Un cop analitzada, podem constatar que l'API està organitzada en els següents serveis:

- *CalendarServiceImpl*
- *ClassroomServiceImpl*
- *MailServiceImpl*
- *PeopleServiceImpl*
- *SubjectServiceImpl*
- *UserServiceImpl*

Disposem doncs de serveis de calendari, d'aules, de correu, de persones, d'assignatures i d'usuaris respectivament.

Podem accedir al seu contingut mitjançant el següent enllaç:

http://denver.uoc.es:8080/webapps/uocapi/apidocs/index.html

3.1.1 Els serveis.

Cada un dels recursos dels diferents serveis ens dóna la següent informació:

- L'*URI* del recurs.
- El model de dades que utilitza.
- Les propietats.
- Els codis d'estat o error.
- Un exemple de crida *GET http*.
- Un exemple de resposta XML.
- Un exemple de resposta JSON.

3.1.2 El model de dades.

L'Open API ens facilita el model de dades. La representació de les dades, que utilitza terminologia basada en *XML Schema*[4], dependrà del consumidor final.

[4] Llenguatge d'esquemes per descriure l'estructura i restriccions dels documents XML.

Ens cal analitzar l'estructura del model de dades per poder tenir una visió estructural dels recursos disponibles mitjançant els diferents serveis que proporciona l'API.

La anàlisi del model ens ha de permetre entendre l'accés a les dades del campus virtual. L'objectiu és visualitzar conceptualment el model de dades, és a dir: la seva distribució, les seves relacions i els seus tipus. A tall d'exemple el següent diagrama de seqüència mostra les peticions *HttpGet* que caldria fer per obtenir les dades d'un missatge que es troba en un recurs de comunicació d'una aula concreta (*Se suposa que disposem del token d'accés a l'API)*:

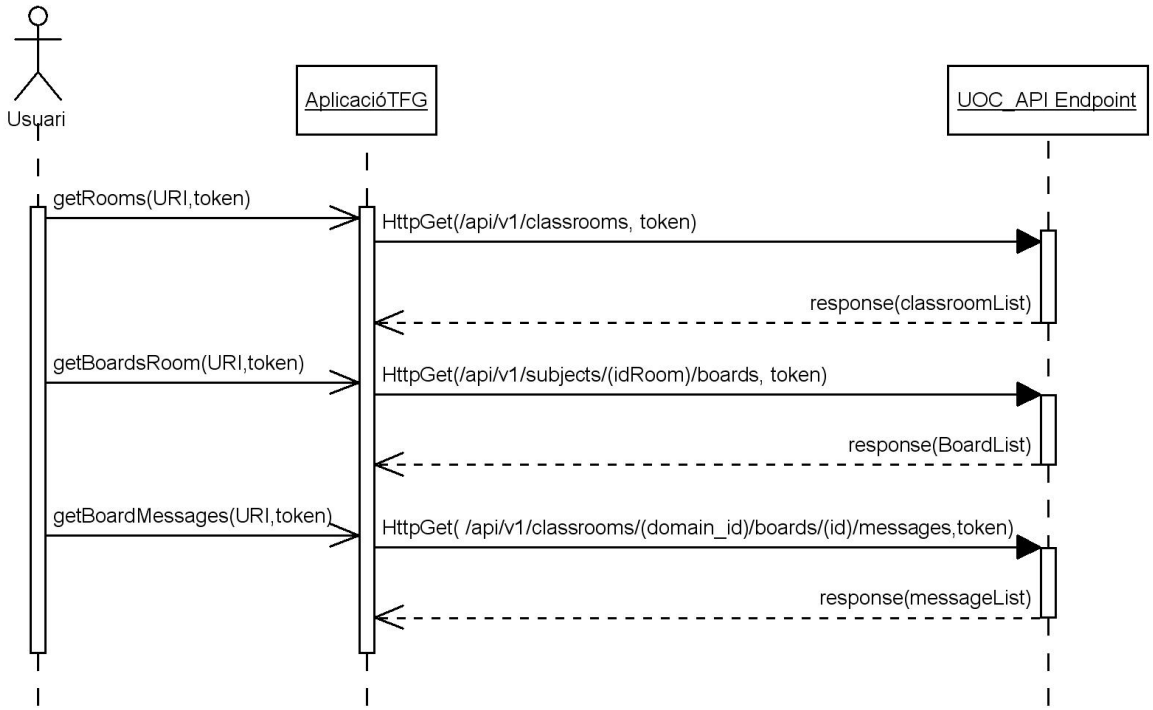

Figura 3 – Peticions HttpGet.

Les dades fonamentals per entendre el model són: l'URI del recurs, els paràmetres, si s'escau, i el tipus de retorn.

Un cop analitzat el model només ens resta implementar les classes Java corresponents que permetin gestionar i emmagatzemar les dades des del llenguatge Java. Aquí en podem trobar exemples, concretament a la secció "*files and libraries*", apartat "*Java JSON client librariy*" de la documentació de *l'Open API*.

3.2 Casos d'ús.

A continuació podem veure el diagrama de casos d'ús del projecte:

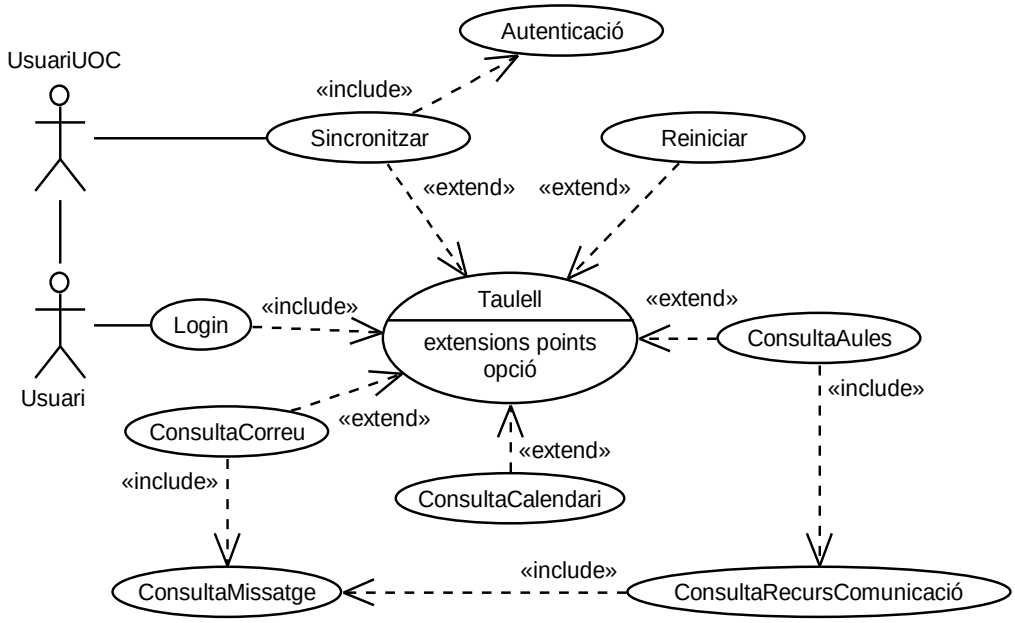

Figura 4 – Casos d'ús.

L'actor *Usuari* s'autentica al programa mitjançant credencials: un usuari i una contrasenya (cas d'ús *Login*). Un cop dins del programa, l'usuari pot interaccionar amb el cas d'ús *Taulell* iniciat pel cas d'ús *Login* quan l'autenticació té èxit. El primer cop que s'executa el programa, per tant encara no s'han llegit mai dades a la UOC, només hi ha disponible l'execució del cas d'ús *Sincronitzar*. Aquest cas d'ús necessita que l'usuari disposi de credencials a la UOC, és a dir, ha de ser un actor *UsuariUOC*.

El cas d'ús *Sincronitzar* es connecta amb la UOC i actualitza la base de dades local. Després retorna automàticament al cas d'ús *Taulell*. Aquest cop, amb totes les opcions del menú disponibles. Les diferents opcions del cas d'ús *Taulell* fan referència a la consulta de les dades descarregades pel cas d'us *Sincronitzar*, com ara consultar les darreres novetats de les aules i els seus recursos (correu, calendari, recursos de comunicació,..).

3.2.1 Documentació textual dels casos d'ús.

La documentació textual dels casos d'ús i les seves funcionalitats s'analitzen amb detall en els subsegüents apartats.

3.2.1.1 Cas d'ús número 1: "Login".

Aquest cas d'ús permet l'autenticació de l'usuari al programa i opcionalment el canvi de contrasenya.

Nom:	**Login**
Resum de la funcionalitat	Permet l'entrada i execució del programa i opcionalment el canvi o configuració de la contrasenya.
Actors	Usuari, UsuariUOC
Casos d'ús relacionats	Taulell
Precondició	L'usuari necessita credencials per autenticar-se al programa, o bé és el primer cop que s'executa l'aplicació i demana la introducció de la contrasenya del programa per primer cop.
Postcondició	L'usuari s'ha autenticat al programa i ha executat el cas d'ús *Taulell* que presenta a pantalla el menú d'opcions, o bé, ha triat l'opció "*sortir*" i el programa ha aturat la seva execució.
Descripció detallada	Si és el primer cop que s'executa el programa demana introduir un usuari i una contrasenya. Altrament, l'usuari introdueix les credencials i entra al programa. Opcionalment pot decidir canviar la contrasenya.

3.2.1.2 Cas d'ús número 2: "Taulell".

Aquest cas d'ús presenta, a l'usuari que ha executat el programa, un menú amb les opcions disponibles.

Nom:	**Taulell**
Resum de la funcionalitat	Presenta el menú d'opcions.
Actors	Usuari
Casos d'ús relacionats	Sincronitzar, Reiniciar, ConsultaCorreu, ConstulaAules, ConsultaCalendari
Precondició	L'usuari a executat el programa.
Postcondició	L'usuari a triat alguna opció del menú i ha aparegut la pantalla que permet gestionar-la, o bé, ha triat l'opció "*sortir*" i el programa ha aturat la seva execució.
Descripció detallada	L'usuari disposa de 6 opcions a triar. Cada una de les opcions es presenten en format botó. L'usuari pot seleccionar l'opció triada prement sobre el botó corresponent. Així doncs des del menú es pot executar qualsevol dels casos d'ús relacionats. Si encara no s'ha fet cap connexió només hi ha activada l'opció "*Sincronitzar*". Altrament, si la darrera sincronització feta té més de 5 dies mostra un avís on s'informa sobre la necessitat de sincronitzar.

3.2.1.3 Cas d'ús número 3: "Reiniciar".

Esborra totes les dades de la BBDD local.

Nom:	**Reiniciar**
Resum de la funcionalitat	Esborra la BBDD local.
Actors	Usuari
Casos d'ús relacionats	Taulell
Precondició	L'usuari es troba davant del menú.
Postcondició	L'usuari ha escollit l'opció "*Reiniciar*"
Descripció detallada	Aquesta opció esborra totes les dades de la base de dades local. Equival a reiniciar el curs, normalment per l'inici d'un nou curs.

3.2.1.4 Cas d'ús número 4: "Sincronitzar".

L'usuari s'autentica a la UOC i concedeix permisos d'accés al programa sobre els recursos del campus. El sistema descarrega localment les darreres novetats del campus i les emmagatzema en una BBDD local.

Nom:	**Sincronitzar**
Resum de la funcionalitat	Autenticació i Connexió al campus per descarregar-se les darreres novetats localment de forma automàtica.
Actors	Usuari
Casos d'ús relacionats	Taulell, Autenticació
Precondició	L'usuari es troba davant del menú.
Postcondició	L'usuari ha triat l'opció "Sincronitzar"
Descripció detallada	1) L'usuari s'autentica a la UOC. 2) L'usuari dóna permisos d'accés i gestió al nostre programa sobre els recursos del campus. 3) El sistema inicia la consulta i descarrega les darreres novetats. 4) S'emmagatzemen les dades en una BBDD local. 5) El programa mostra un missatge confirmant l'èxit de la sincronització i després retorna al cas d'ús Taulell.
Esdeveniments alternatius	L'usuari tria l'opció "Sortir" i retorna al cas d'ús Taulell.
Excepcions	En el pas 1) L'usuari no té autorització. El *token* ha expirat o bé la connexió amb el campus no respon. El programa mostra un missatge informant de la situació i es roman a la pantalla d'autenticació. En el pas 4) El sistema no pot obrir o connectar-se amb la BBDD. Mostra un missatge informant-ne i retorna al cas d'ús Taulell.

3.2.1.5 Cas d'ús número 5: "Autenticació".

Permet autenticar-se a la UOC i concedir permisos al programa per utilitzar l'API.

Nom:	**Autenticació**
Resum de la funcionalitat	Autenticar-se a la UOC i concedir permisos per utilitzar l'API.
Actors	Usuari
Casos d'ús relacionats	Taulell, Sincronitzar
Precondició	L'usuari té credencials d'autenticació a la UOC. És un estudiant.
Postcondició	S'han descarregat localment les darreres novetats del portal virtual de l'usuari a la UOC.
Descripció detallada	1) L'usuari introdueix les credencials de la UOC. 2) L'usuari concedeix permisos per accedir a les dades del seu taulell a través de l'API. 3) El sistema mostra un missatge i retorna al cas d'ús 1) Taulell.
Excepcions	En el pas 1) L'usuari no és correcte, no té autorització o bé la connexió amb el campus no respon. El programa mostra un missatge informant de la situació i es manté a la pantalla d'autenticació. En el pas 3) El *token* ha expirat o bé el sistema no pot obrir o connectar-se amb la BBDD. Mostra un missatge informant-ne i es manté a la pantalla d'autenticació o bé retorna al cas d'ús Taulell respectivament.

3.2.1.6 Cas d'ús número 6: "ConsultaCorreu".

Permet consultar les carpetes disponibles del correu de l'usuari i els seus missatges nous o bé tots els missatges.

Nom:	**ConsultaCorreu**
Resum de la funcionalitat	Permet consultar tots els missatges o només els nous missatges dels diferents recursos del correu de l'usuari.
Actors	Usuari
Casos d'ús relacionats	Taulell, ConsultaMissatge
Precondició	L'usuari ha actualitzat les dades en algun moment i disposa d'accés a la BBDD.
Postcondició	S'han consultat els recursos del correu.
Descripció detallada	1) L'usuari selecciona el recurs de correu que vol consultar. 2) L'usuari selecciona, entre tots els missatges o entre els missatges nous, el missatge que vol consultar. 3) L'usuari prem el botó acceptar i executa el cas d'ús *ConsultaMissatge*.
Esdeveniments alternatius	L'usuari tria l'opció "Sortir" i retorna al cas d'ús Taulell.
Excepcions	No hi ha accés a la BBDD, cas en que mostra un missatge i es manté al mateix cas d'ús.

3.2.1.7 Cas d'ús número 7: "ConsultaMissatge".

Permet consultar un missatge d'un recurs de comunicació prèviament seleccionat.

Nom:	**ConsultaMissatge**
Resum de la funcionalitat	Permet consultar un missatge d'un recurs de comunicació.
Actors	Usuari
Casos d'ús relacionats	ConsultaCorreu, ConsultaRecursComunicacio.
Precondició	L'usuari ha seleccionat un recurs de comunicació i en vol consultar un missatge concret.
Postcondició	S'ha consultat un missatge d'un recurs de comunicació.
Descripció detallada	L'usuari consulta un missatge corresponent a un recurs de comunicació seleccionat anteriorment.
Esdeveniments alternatius	L'usuari tria l'opció "Sortir" i retorna al cas d'ús *ConsultaCorreu* o bé al cas d'us *ConsultaRecursComunicacio* depenent del seu cas d'ús antecessor.
Excepcions	No hi ha accés a la BBDD, cas en que mostra un missatge i es manté al mateix cas d'ús.

3.2.1.8 Cas d'ús número 8: "ConsultaCalendari".

Permet consultar les darreres novetats del calendari.

Nom:	**ConsultaCalendari**
Resum de la funcionalitat	Permet consultar els darrers esdeveniments o novetats del calendari.
Actors	Usuari
Casos d'ús relacionats	Taulell
Precondició	L'usuari ha actualitzat les dades en algun moment i disposa d'accés a la BBDD.
Postcondició	S'han consultat els esdeveniments del calendari.
Descripció detallada	L'usuari selecciona la data del calendari de la qual vol consultar els esdeveniments.
Esdeveniments alternatius	L'usuari tria l'opció "Sortir" i retorna al cas d'ús Taulell.
Excepcions	No hi ha accés a la BBDD, cas en que mostra un missatge i es manté al mateix cas d'ús.

3.2.1.9 Cas d'ús número 9: "ConsultaAules".

Permet consultar les darreres novetats d'una aula.

Nom:	**ConsultaAules**
Resum de la funcionalitat	Permet consultar les darreres novetats d'una aula.
Actors	Usuari
Casos d'ús relacionats	Taulell, ConsultaRecursComunicacio, ConsultaMissatge
Precondició	L'usuari ha actualitzat les dades en algun moment i té d'accés a la BBDD.
Postcondició	S'han consultat els recursos d'una Aula.
Descripció detallada	1) L'usuari selecciona una aula. 2) L'usuari selecciona un recurs de comunicació de l'aula. 3) L'usuari prem el botó acceptar i executa el cas d'ús *ConsultaRecursComunicacio*.
Esdeveniments alternatius	L'usuari tria l'opció "Sortir" i retorna al cas d'ús Taulell.
Excepcions	No hi ha accés a la BBDD, cas en que mostra un missatge i es manté al mateix cas d'ús.

3.2.1.10 Cas d'ús número 10: "ConsultaRecursComunicacio".

Permet consultar els missatges d'un recurs de comunicació d'una aula concreta.

Nom:	**ConsultaRecursComunicacio**
Resum de la funcionalitat	Permet consultar les darreres novetats d'un recurs de comunicació.
Actors	Usuari
Casos d'ús relacionats	ConsultaAules, ConsultaMissatge
Precondició	L'usuari ha actualitzat les dades en algun moment i disposa d'accés a la BBDD. A més ha seleccionat una aula que té algun recurs de comunicació
Postcondició	S'han consultat els recursos de comunicació de l'aula seleccionada.
Descripció detallada	1)L'usuari selecciona, entre tots els missatges o només entre els missatges nous del recurs de comunicació, el missatge que vol consultar. 2)L'usuari prem el botó "Acceptar" i executa el cas d'ús *ConsultaMissatge*
Esdeveniments alternatius	L'usuari tria l'opció "sortir" i retorna al cas d'ús *ConsultaAules*.
Excepcions	No hi ha accés a la BBDD, cas en que mostra un missatge i es manté al mateix cas d'ús. L'usuari prem el botó "Cancel·lar" i retorna al cas d'us *ConsultaAules*.

3.3 El diagrama estàtic d'anàlisi.

El model estàtic d'anàlisi és aquell en què es descriuen les classes i els objectes. S'anomena estàtic perquè mostra totes les relacions possibles al llarg del temps i no quines són vàlides en un cert moment.

El model estàtic utilitza el diagrama de classes per representar-lo. Un diagrama de classes mostra l'estructura estàtica de les classes en un domini (porció del món real considerada per una aplicació); s'hi mostren les classes i les relacions entre aquestes, que poden ser d'herència, d'associació, d'agregació o d'ús.

3.3.1 El diagrama de classes.

El següent diagrama de classes representa el domini de la nostra aplicació (usuaris (estudiants i tutors), semestres, classes, *events* o esdeveniments, carpetes, missatges i recursos de comunicació):

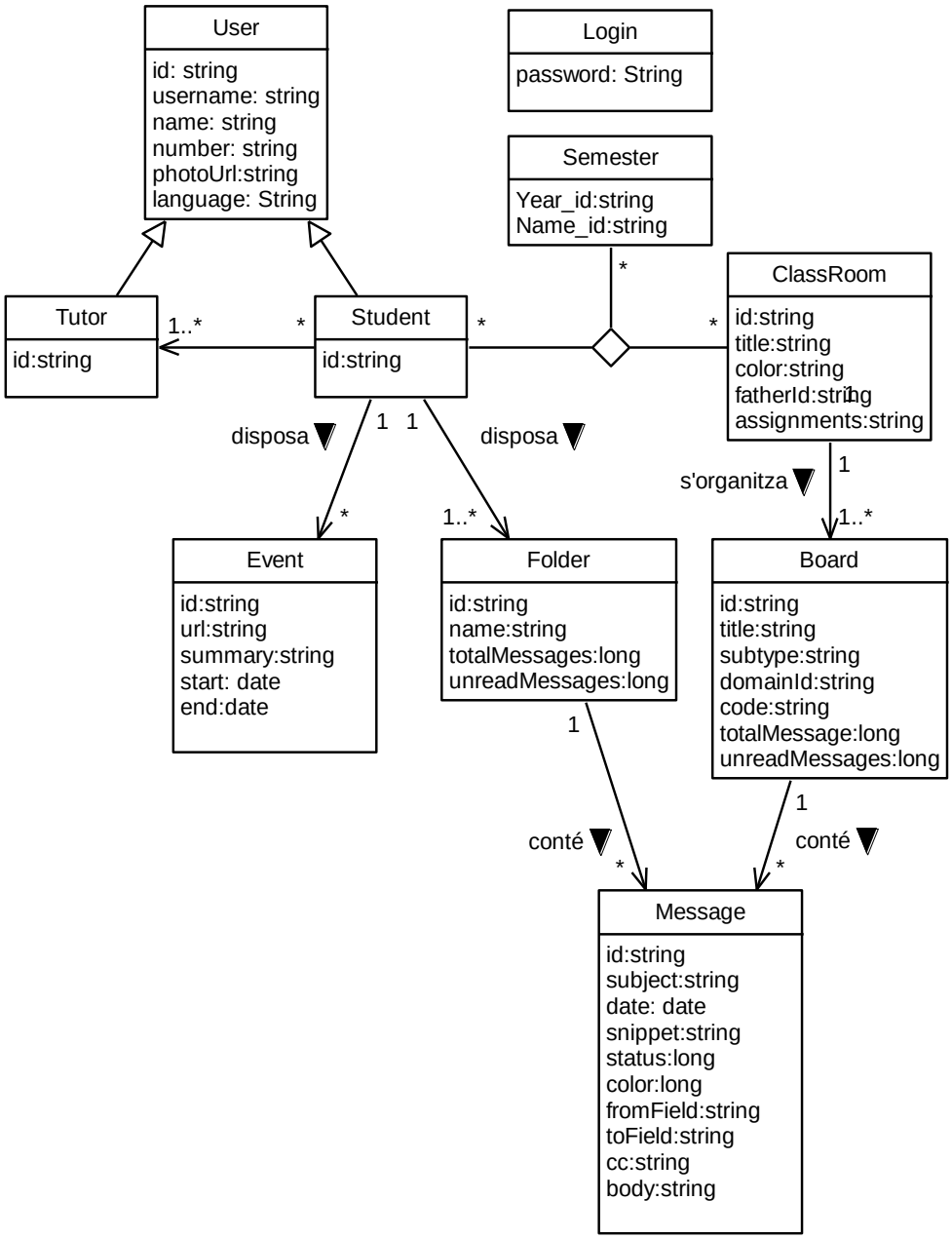

Figura 5 – Diagrama de classes

Les classes *Tutor* i *Student* hereten *d'User*. Totes dues representen un usuari del sistema.

La classe *Student* disposa d'esdeveniments (*events*, potser cap *) al calendari i de carpetes de correu (*Folder*) que al seu torn contenen missatges (potser cap *).

Hi ha una relació ternària entre les classes *Student, Semester i ClassRoom*. Un estudiant en un semestre determinat pot estar matriculat de moltes assignatures. Una assignatura en un determinat semestre pot tenir molts estudiants matriculats. Un estudiant pot estar matriculat d'una assignatura en molts semestres. Això permet que un estudiant pugui matricular-se d'assignatures suspeses.

La classe *Semester* emmagatzema l'any i el nom del semestre que pot ser: tardor o primavera.

Una assignatura (*ClassRoom*) pot tenir molts recursos de comunicació dins d'una aula. Al seu torn cada recurs de comunicació pot tenir molts missatges o potser cap (*).

La classe *Login* permet emmagatzemar la contrasenya d'entrada al programa.

4. Disseny.

4.1 Arquitectura del programari.

Un cop establerts els requeriments del projecte, hem de definir l'arquitectura en la qual es basarà el sistema. Aquesta decisió afectarà a la resta de decisions de disseny, a l'estructura, i al funcionament del programa.

4.1.1 Arquitectura en capes o nivells.

"Les arquitectures en capes o nivells (layered architectures) representen una organització jeràrquica dels elements d'un sistema, de manera que cada capa proporciona serveis als elements de la capa immediatament anterior, i se serveix dels serveis que li brinden els elements de la capa immediatament següent." (Moreno Vergara, N., Vallecillo Moreno, A., Romero Salguero, J.R., Durán Muñoz, F.)

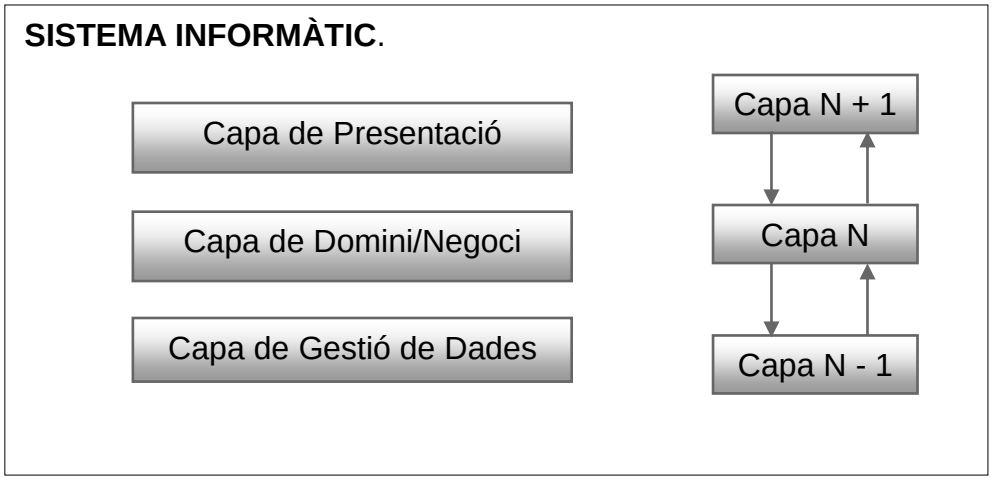

Figura 6 – Arquitectura en capes.

L'arquitectura en capes permet la descomposició d'un problema complex en subsistemes independents i complementaris. Permet **optimitzacions** i **refinaments**.

A més, facilita la **reutilització**. Si mantenim la mateixa interfície podem tenir diferents versions de cada capa. Això facilita la **portabilitat**. Per portar el sistema a un entorn diferent, n'hi ha prou d'implementar de nou aquest nivell.

4.1.2 Nivell lògic.

En aquest apartat representem el nivell lògic de les diferents capes del model mitjançant paquets.

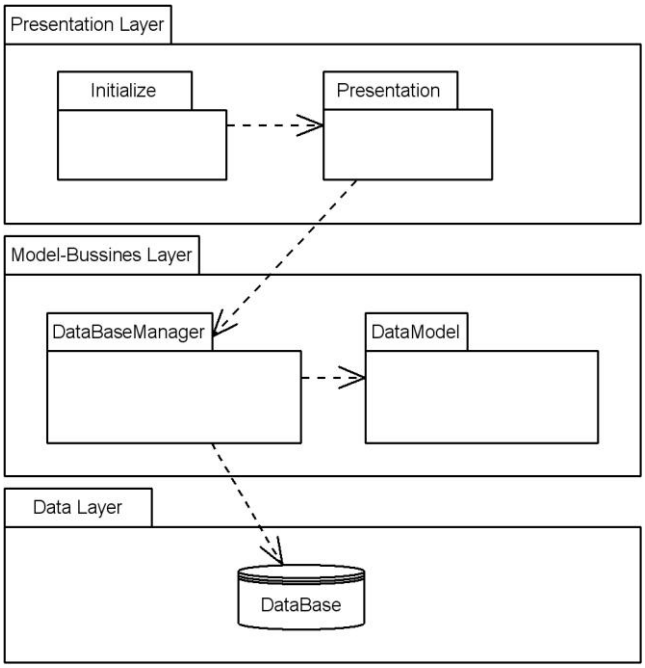

Figura 7 – Nivell lògic.

La capa *Presentation Layer* és la responsable d'executar l'aplicació i atendre, mitjançant una classe controladora, les diferents peticions de les classes frontera del *package Presentation* que formen la interfície gràfica d'usuari.

4.1.3 El patró Façana.

Per comunicar les diferents capes aplicarem el patró façana que permet, de forma centralitzada, la comunicació d'una capa amb la seva capa adjacent. Per aconseguir-ho hem d'afegir una nova classe façana que representa la capa o subsistema accedit. El patró façana concentra la dependència entre capes en un únic punt. Això permet reduir l'acoblament i facilita els canvis que tindran un mínim impacte sobre la resta de capes.

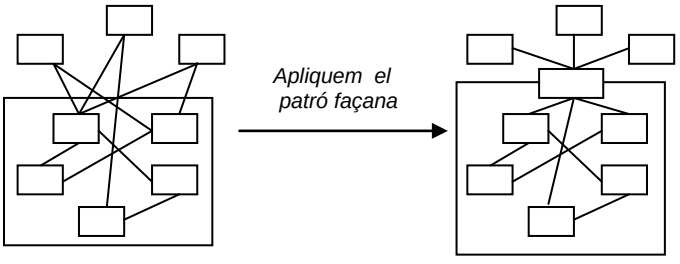

Figura 8 – El patró façana

4.1.3.1 La capa de Presentació

Aquesta capa es l'encarregada d'interactuar amb l'usuari, normalment, mitjançant una interfície gràfica. La capa de presentació conté els components necessaris per interactuar amb l'usuari de l'aplicació. Les responsabilitzades d'aquesta capa són: obtenir i mostrar dades a l'usuari, validar-les i permetre la interacció d'aquesta capa amb la resta del sistema.

4.1.3.2 El patró Model Vista Controlador (MVC).

A la capa de presentació aplicarem el patró d'arquitectura *Model-Vista-Controlador* en la seva variant adaptada per a l'arquitectura en capes. Dividirem les classes que representen pantalles en models, vistes i controladors.

Tot seguit podem veure el diagrama de col·laboració associat a l'acció de consultar les aules d'un estudiant quan ens trobem a la pantalla "*Taulell*":

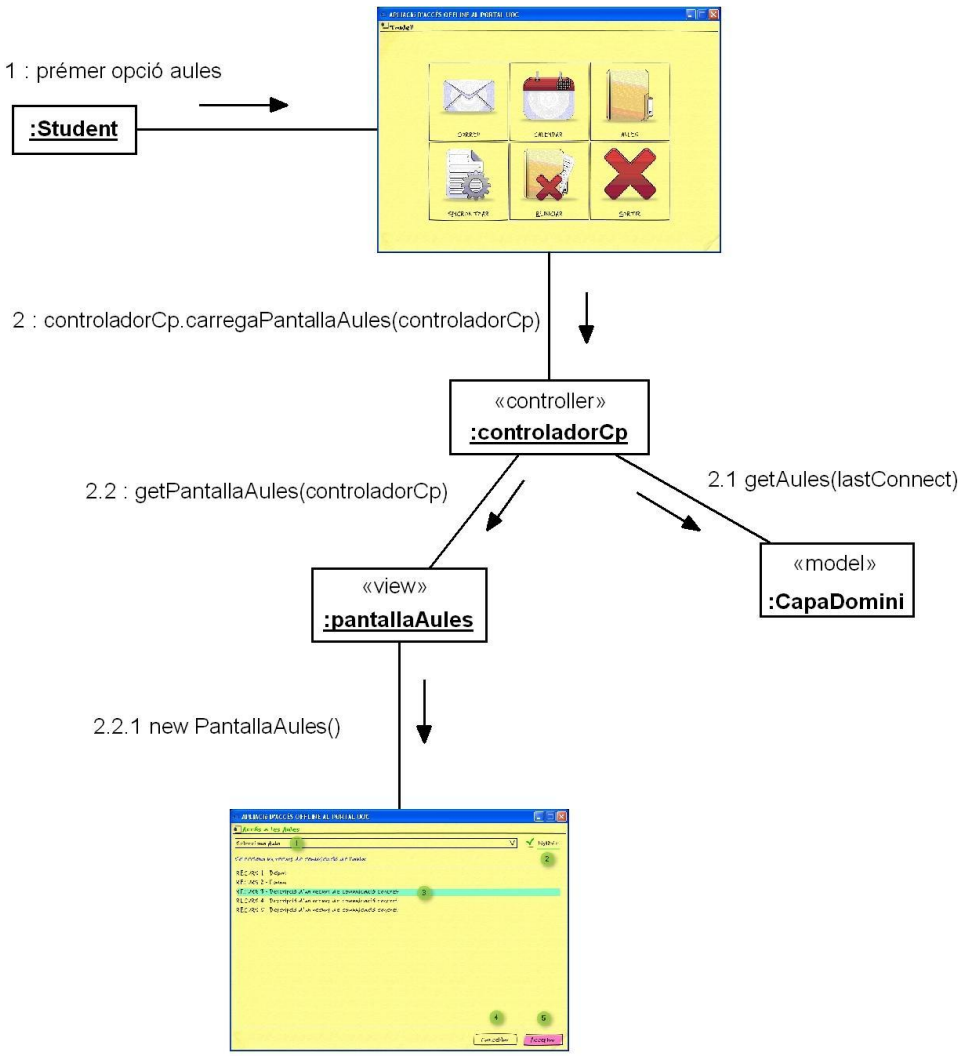

Figura 9 – Diagrama de col·laboració de l'acció "consultar aules".

A la capa de presentació fem servir una classe *controlador* que atén totes les peticions de les classes vista que representen pantalles. Un exemple seria la classe *pantallaAules*. Finalment, el model serà la capa de domini.

A cada pantalla de l'aplicació li associarem una classe *controlador* que gestionarà les peticions dels objectes interns de la pantalla. Per tant, tindrem una vista i un controlador per cada classe de tipus pantalla.

4.2 Prototipatge.

En aquest apartat analitzarem la interfície gràfica d'usuari tot presentant-ne un prototip de pantalles que s'ajusten a les descripcions definides als casos d'ús de l'apartat 3.2 d'aquest mateix document. Totes les dades que ofereix el programa es consulten fora de línia.

4.2.1 Pantalles cas d'ús *Login*.

En realitat el cas d'us *Login* està composat per tres pantalles: **Pantalla de Benvinguda, Pantalla d'entrada de contrasenya i Pantalla de canvi de contrasenya**.

El primer cop que s'executa l'aplicació no hi ha cap contrasenya. El programa ho detecta i mostra un missatge de benvinguda. Després demana introduir una contrasenya per protegir les properes execucions del programa. Les pantalles són:

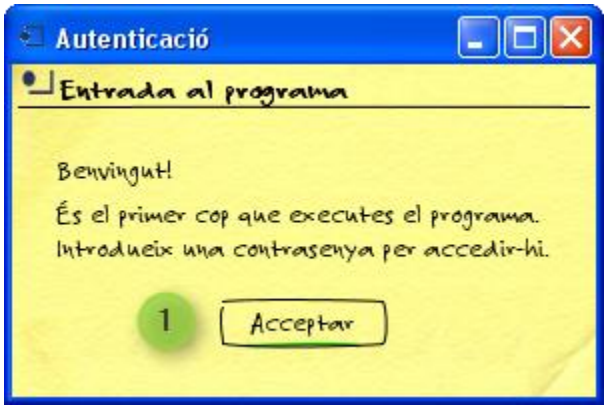

Figura 10 – Pantalla de Benvinguda.

Figura 11 – Pantalla de contrasenya inicial.

Les opcions (1) i (2) han de permetre introduir una contrasenya amb robustesa.

Les següents execucions demanaran directament la contrasenya per entrar al programa. També permetrà l'opció de canviar la contrasenya. Vegem-ne les pantalles:

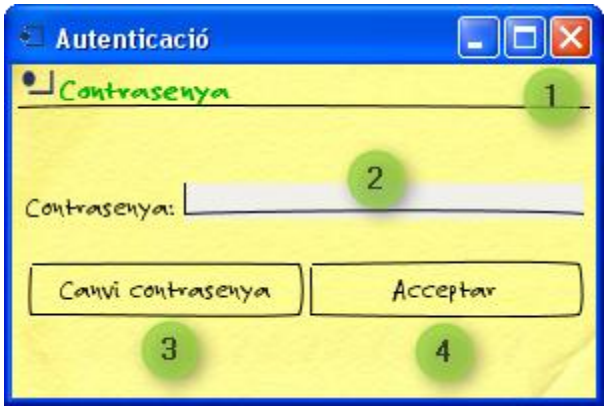

Figura 12 – Pantalla d'entrada al programa.

Aquesta pantalla també ha de permetre introduir la contrasenya amb robustesa.

Si el que volem es canviar la contrasenya actual hem de prémer el boto "Canviar contrasenya". La següent pantalla permetrà el canvi de contrasenya:

Figura 13 – Canvi de contrasenya.

En el punt (1) ens demana que introduïm la contrasenya actual. En el punt (2) hem d'introduir la nova contrasenya i en el punt (3) tornar a escriure-la per seguretat. El canvi de contrasenya també hauria de ser robust.

El primer cop que s'entra al programa, la pantalla "**Taulell**" només hauria de deixa executar les opcions "**Sincronitzar**" o bé "**Sortir**", donat que encara no hi ha dades per consultar. La pantalla seria la següent:

Figura 14 – Pantalla Taulell – primera execució.

Només hi ha disponibles les opcions de "**Sincronitzar**" i "**Sortir**". Per poder executar l'opció sincronitzar cal tenir credencials d'usuari a la UOC. Si no s'executa l'opció *sincronitzar* la resta d'opcions no estaran disponibles.

4.2.2 Pantalla Taulell.

Aquesta pantalla es correspon amb el cas d'ús número 2 i presenta el menú principal de l'aplicació:

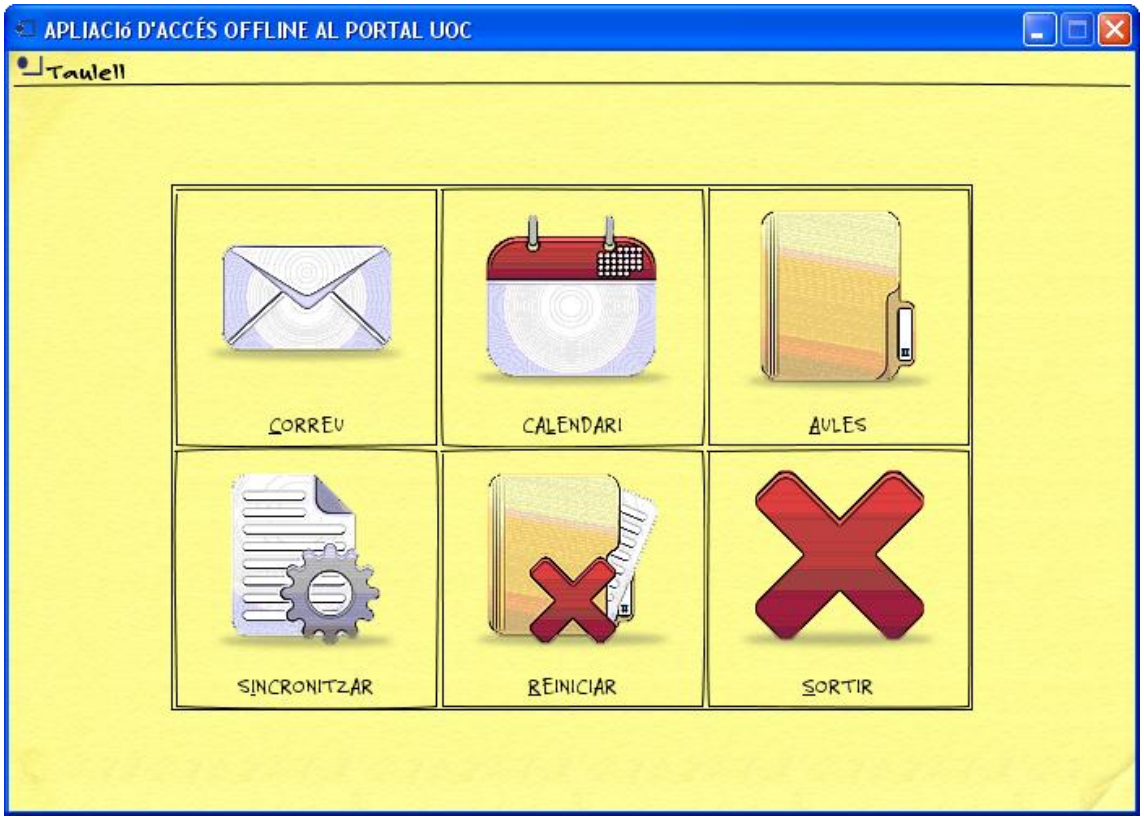

Figura 15 – Pantalla Taulell.

Cada opció està associada amb un cas d'ús:

- L'opció Correu executa el cas d'ús número 5 *ConsultaCorreu*.
- L'opció Calendari executa el cas d'us número 7 *ConsultaCalendari*.
- L'opció Aules executa el cas d'ús número 8 *ConsultaAules.*
- L'opció Sincronitzar executa el cas d'us número 3 *Sincronitzar*.
- L'opció Reiniciar executa el cas d'us número 2 *Reiniciar*.
- L'opció Sortir surt del programa.

Nota: Les dades que podem consultar es corresponen amb les del darrer usuari que ha executat l'opció sincronitzar.

4.2.3 Pantalla Correu.

Aquesta pantalla es correspon amb el cas d'ús número 6 i permet consultar les novetats del correu de l'usuari.

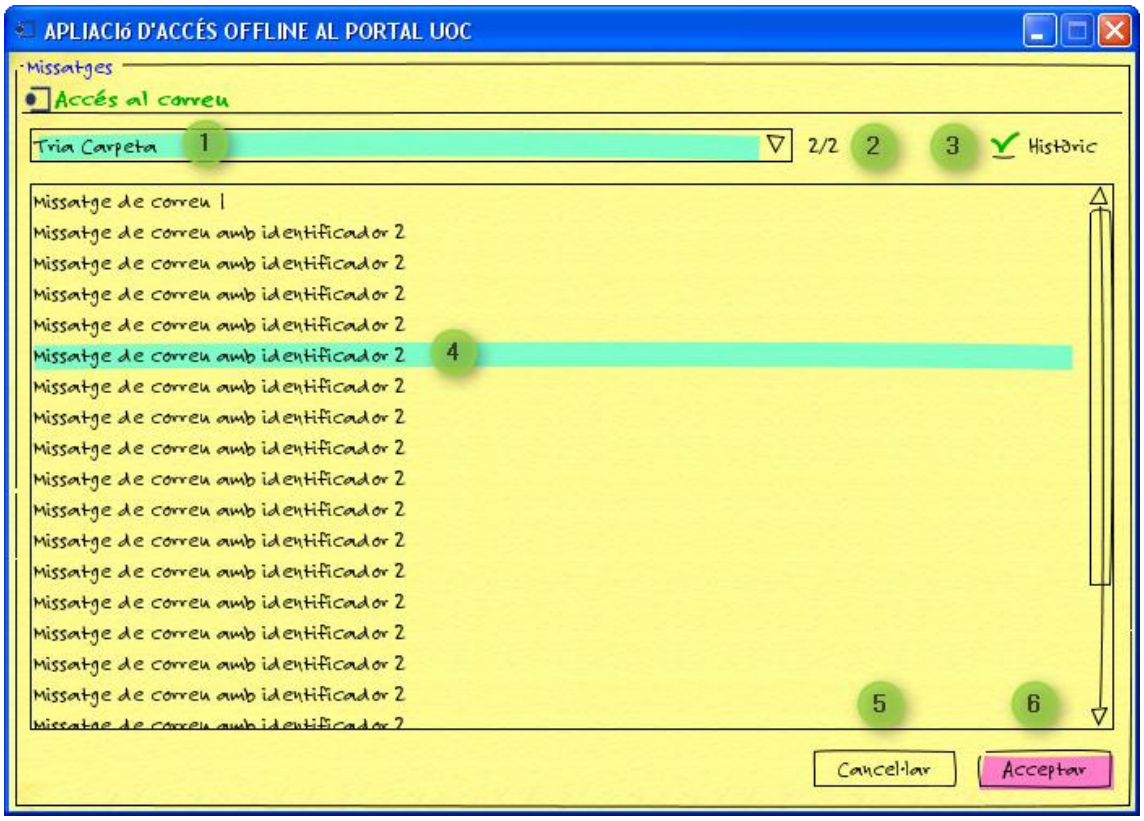

Figura 16 – Pantalla Correu.

- El desplegable (**1**) permet triar una carpeta del correu. Un cop triada, el punt (**2**) mostra els missatges nous respecte el total.
- El vist-i-plau "*Històric*" (**3**) permet tenir en compte l'històric de missatges. Si està seleccionat mostra els missatges nous. Altrament els mostra tots.
- La llista (**4**) permet visualitzar els missatges de la carpeta de correu seleccionada.
- Els botons "*Acceptar*" (**5**) i "*Cancel·lar*"(**6**) permeten visualitzar el missatge seleccionat executant el cas d'ús número 6 "*ConsultaMissatge*", o bé retornar a la pantalla Taulell respectivament.

4.2.4 Pantalla Consulta de Missatge.

Aquesta pantalla es correspon amb el cas d'ús número 7 i permet consultar el contingut d'un missatge de correu.

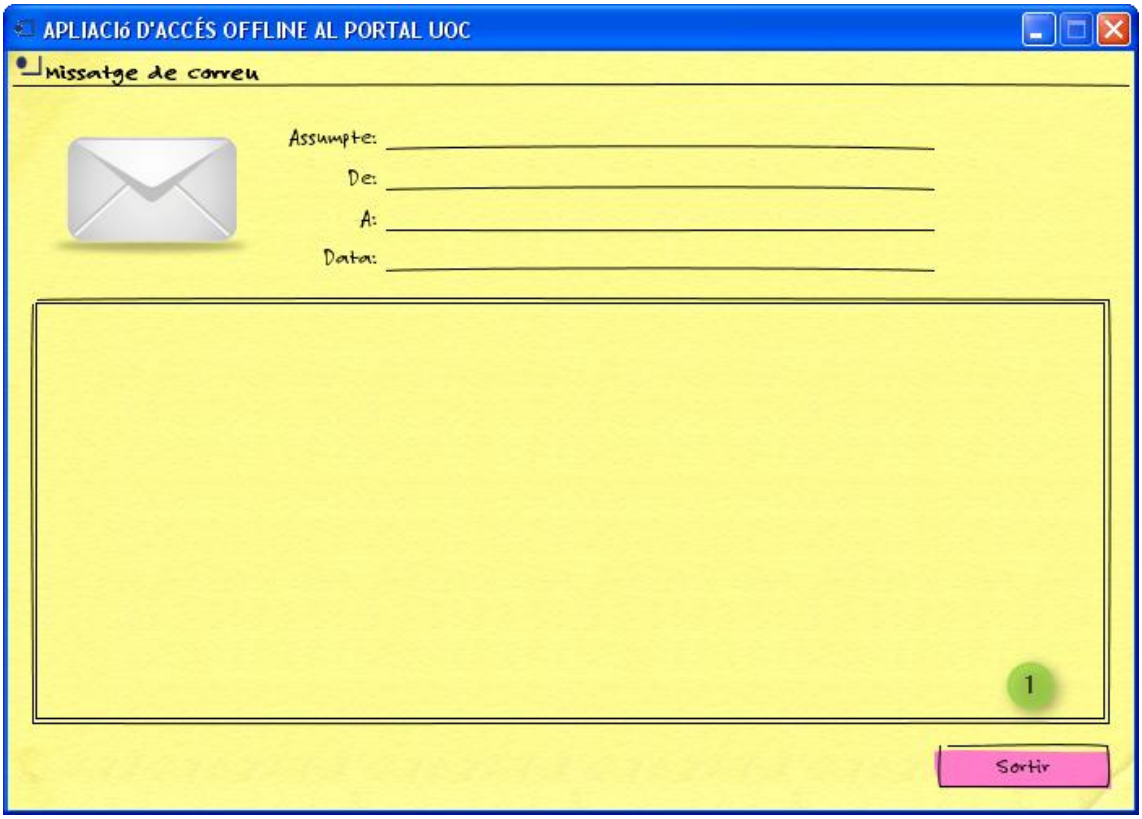

Figura 17 – Pantalla consulta missatge de correu.

- Aquesta pantalla mostra totes les dades d'un missatge: *Assumpte, de, a, data i cos del missatge*.
- Per sortir de la pantalla tenim el botó "*Sortir*" (**1**) que retorna a la pantalla "*Consulta Correu*" que es correspon amb el cas d'ús número 5.

4.2.5 Pantalla Calendari.

Aquesta pantalla es correspon amb el cas d'ús número 8 i permet consultar en el calendari la llista d'esdeveniments i els dies implicats.

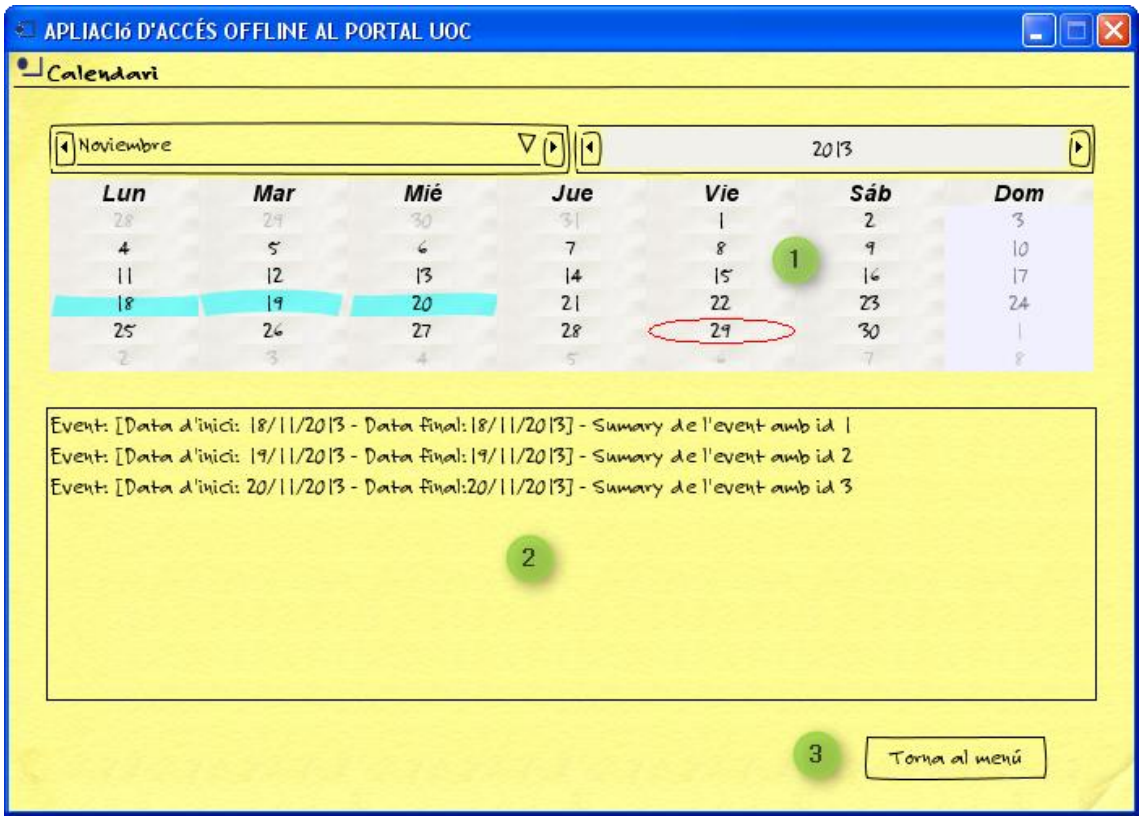

Figura 18 – Pantalla de consulta del calendari

- Aquesta pantalla consta d'un calendari (**1**) situat al mes actual i una llista (**2**) que mostra els esdeveniments d'aquest mes.
- El calendari ressalta amb color blau els dies que tenen algun esdeveniment. El dia actual està encerclat en vermell.
- Si premem sobre el botó "*Sortir*" (**3**) retornem a la pantalla *Taulell*.

4.2.6 Pantalla consulta de les Aules.

Aquesta pantalla es correspon amb el cas d'ús número 10 i permet consultar els recursos corresponents a una aula.

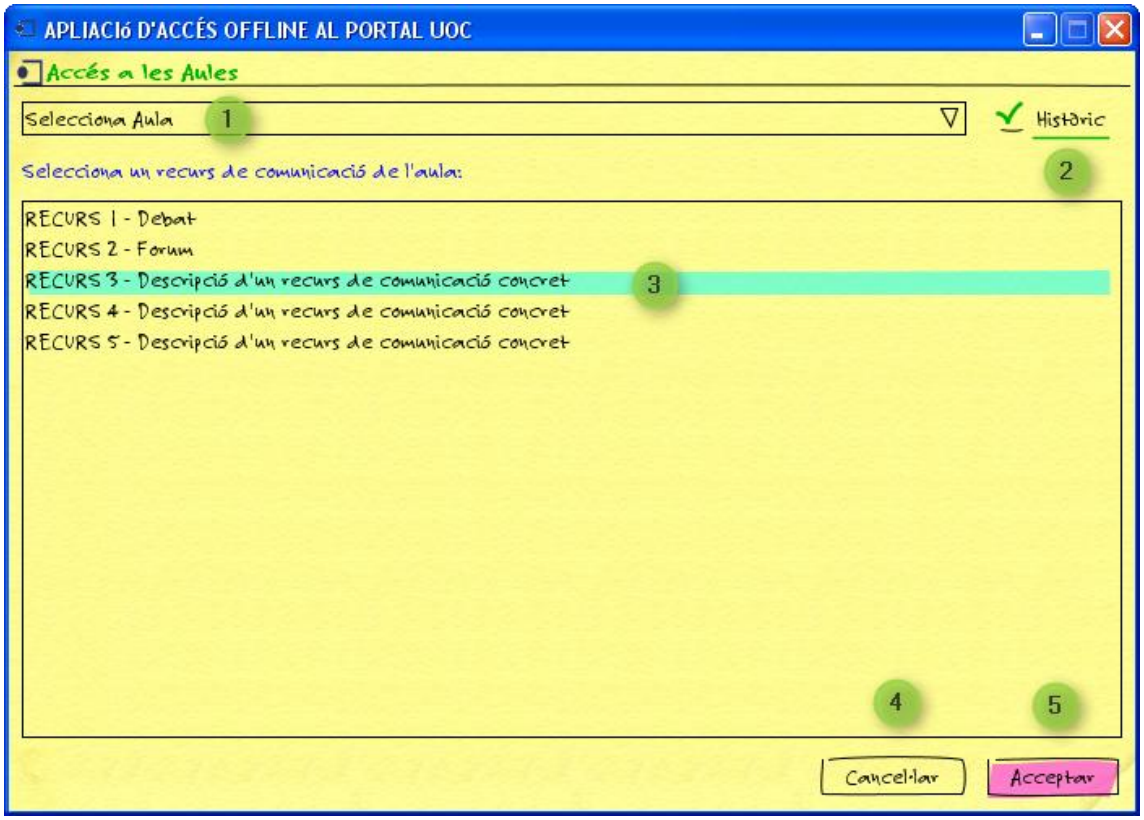

Figura 19 – Pantalla de consulta d'Aules.

- Aquesta pantalla permet seleccionar una aula amb el desplegable (1).
- L'àrea de text (3) permet mostrar els recursos de comunicació disponibles a l'aula seleccionada.
- El vist-i-plau "*Històric*" (2) permet filtrar entre les aules del curs actual, cas no seleccionat, o bé mostrar l'històric de les aules, cas seleccionat.
- Si premem el botó "*Cancel·lar*" (4) retornem a la pantalla *Taulell*. Altrament, si premem el botó "*Acceptar*" (5) s'executa el cas d'ús número 8 "*ConsultaRecursComunicacio*".

4.2.7 Pantalla consulta dels missatges d'un recurs de comunicació.

Aquesta pantalla es correspon amb el cas d'ús número 10 i permet consultar la llista de missatges d'un recurs de comunicació d'una aula.

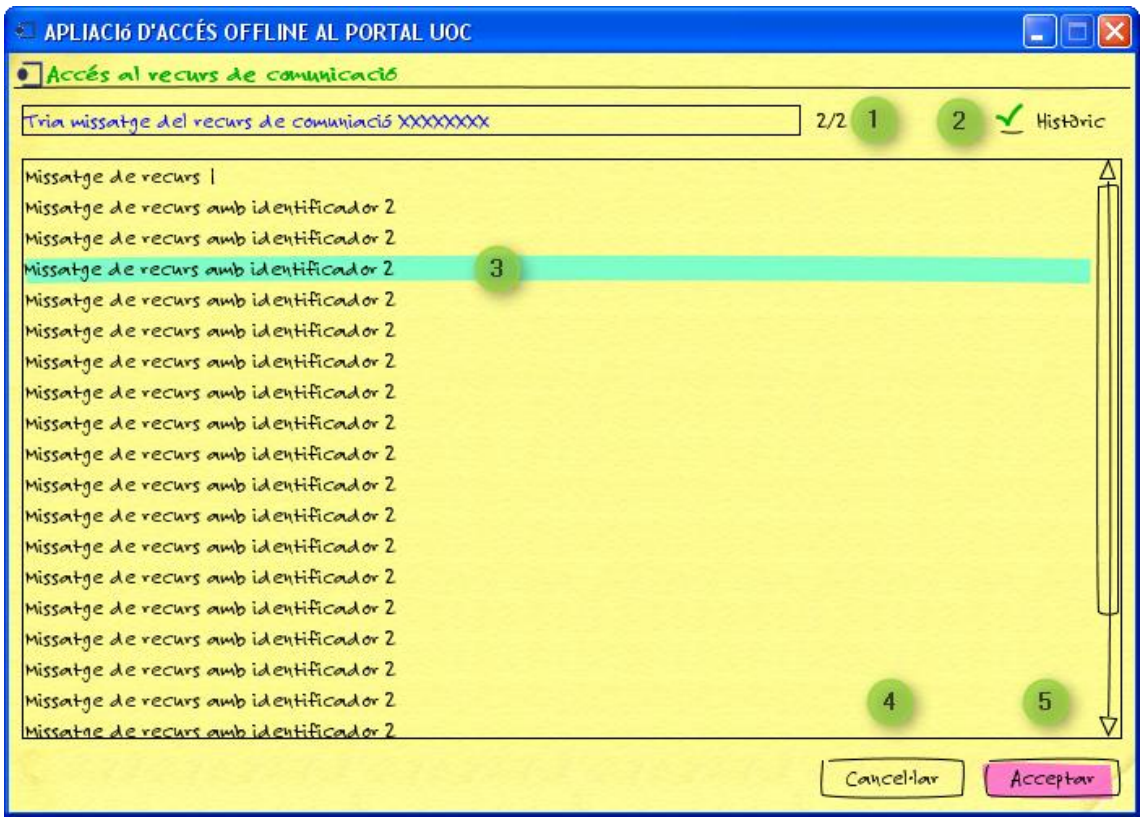

Figura 20 – Pantalla missatges d'un recurs de comunicació.

- Aquesta pantalla permet seleccionar un missatge d'una llista de missatges (3).
- L'etiqueta (1) mostra els missatges nous respecte el total de missatges.
- El vist-i-plau "*Històric*" (2) permet filtrar tots els missatges del recurs (cas seleccionat), o bé, només els missatges nous (cas no seleccionat).
- Si premem el botó "*Acceptar*" s'executa el cas d'ús (6) consulta Missatge. Altrament el botó "*Canel·lar*" retorna a la pantalla Consulta d'Aules.

4.2.8 Pantalla consulta d'un Missatge d'un recurs de comunicació.

Aquesta pantalla es correspon amb el cas d'ús número 7 i permet consultar el contingut d'un missatge d'un recurs de comunicació d'una aula.

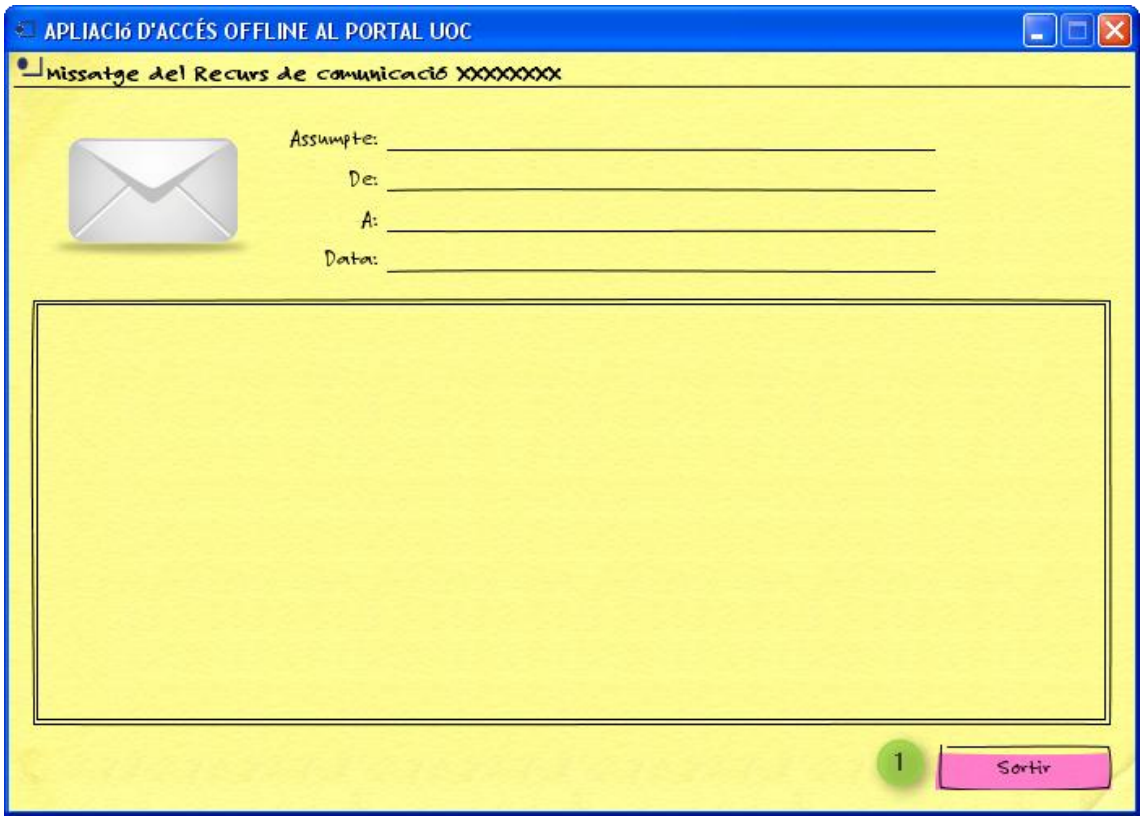

Figura 21 – Pantalla de consulta d'un missatge d'un recurs.

- Aquesta pantalla mostra totes les dades d'un missatge: *Assumpte, de, a, data i cos del missatge*.
- Per sortir de la pantalla tenim el botó "*Sortir*" (**1**) que retorna a la pantalla "*Consulta de missatges d'un recurs de comunicació*" que es correspon amb el cas d'ús número 9.

4.2.9 Pantalla Sincronitzar.

Aquesta pantalla es correspon amb els casos d'ús números 4 i 5 i permet autenticar-se a la UOC i concedir permisos al programa per utilitzar l'API.

Figura 22 – Autenticar-se a la UOC.

Pantalla d'autenticació on hem d'introduir el nom (1) i la paraula de pas (2). Després cal prémer el botó (3) per iniciar sessió, o bé, el botó (4) per netejar el formulari.

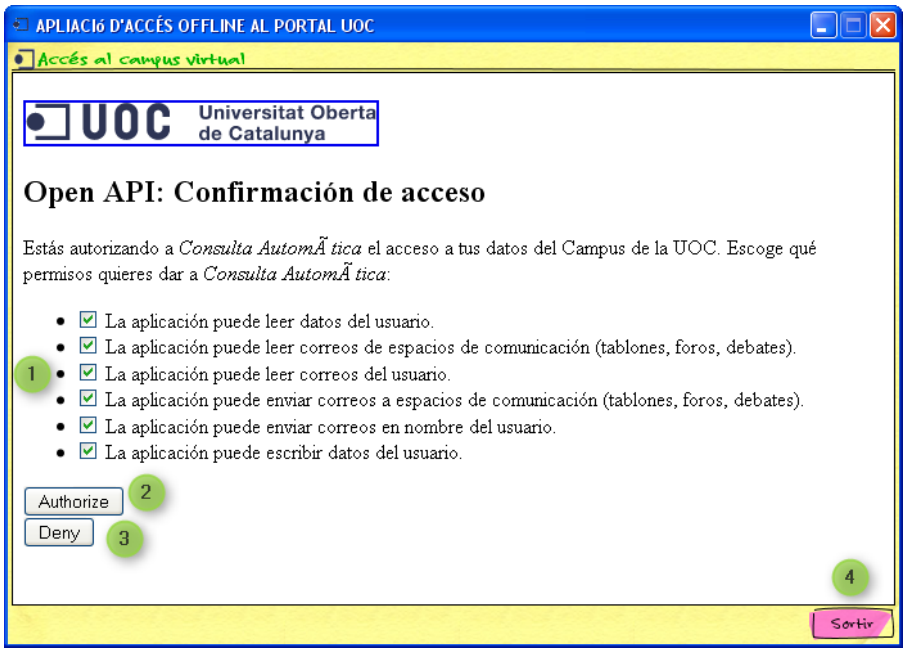

Figura 23 – Pantalla concedir permisos.

Concedir permisos (1) i confirmar l'autorització (2) o denegar-la (3).

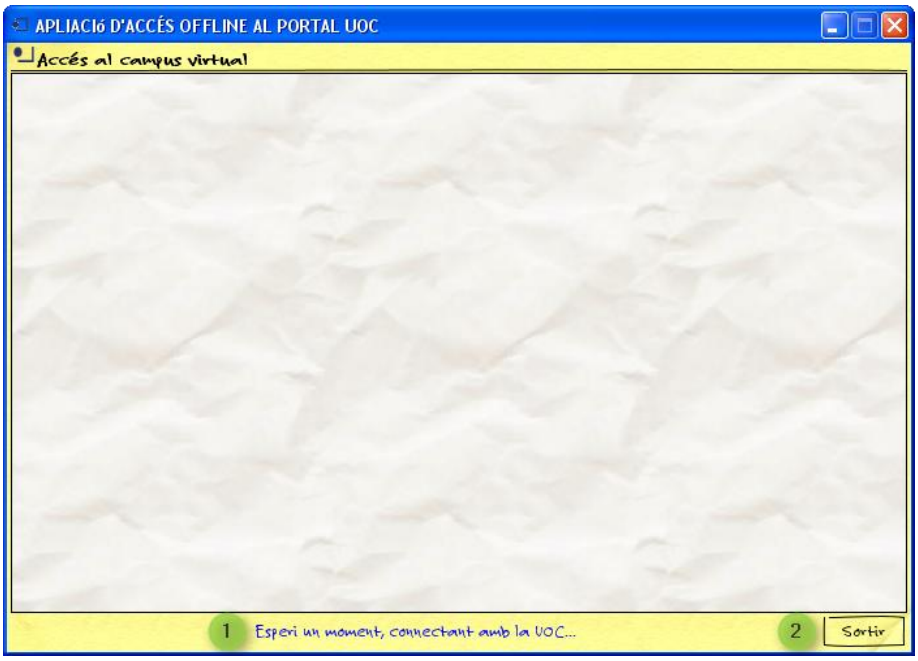

Figura 24 – Pantalla connectant amb la UOC...

Aquesta pantalla mostrarà un missatge (1) mentre s'estableix connexió amb la UOC. Opcionalment, podem sortir (2). Val a dir, que en aquest darrer cas es mostrarà un missatge informant que les dades no s'han actualitzat. En tot cas ens retorna al menú i tindrem l'opció de tornar a intentar-ho.

Un cop establerta la connexió se'ns mostrarà una barra de progrés (1) que ens informa sobre la proporció de dades actualitzades:

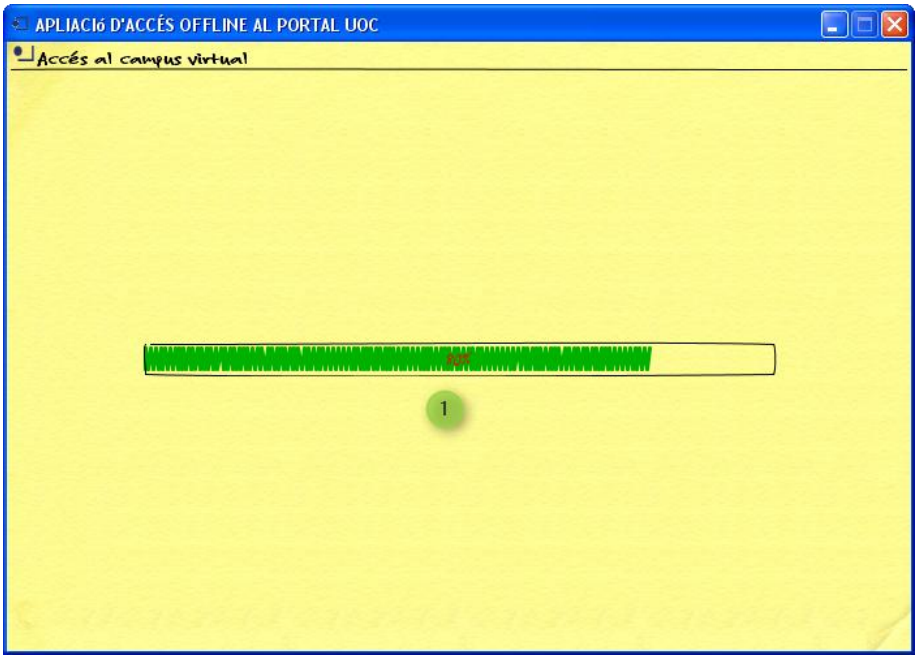

Figura 25 – Barra de progrés de les actualitzacions.

Un cop feta la connexió i actualitzades les dades correctament, es mostrarà el següent missatge confirmant que les dades s'han actualitzat correctament. Cal prémer el botó "Acceptar".

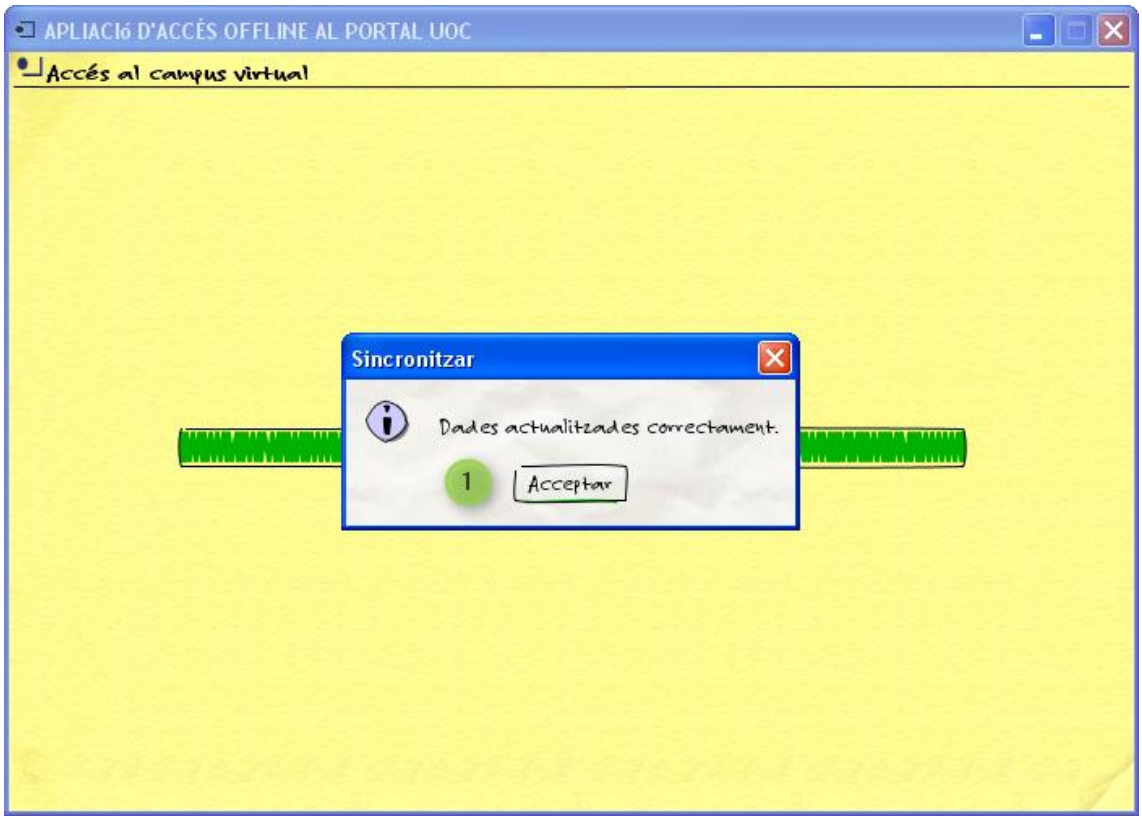

Figura 26 – Pantalla missatge d'actualització de dades.

4.2.10 Pantalla Reiniciar.

Aquesta pantalla es correspon amb els casos d'ús números 3 i permet esborrar totes les dades locals.

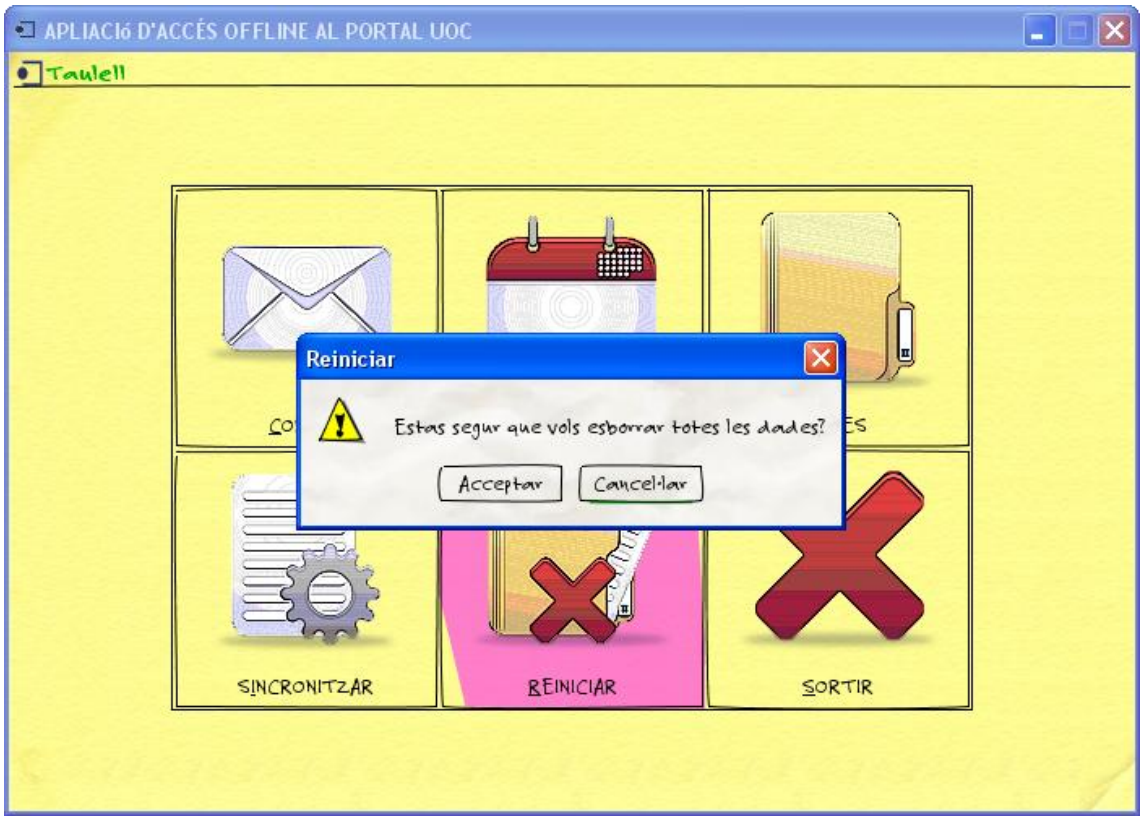

Figura 27 – Pantalla reiniciar dades.

- Quan seleccionem l'opció reiniciar, ens mostra un missatge d'advertència i demana confirmació per esborrar totes les dades.
- Si premem l'opció *acceptar*, s'esborren totes les dades locals i es retorna a la pantalla Taulell. En cas contrari, si premem el botó "*Cancel·lar*" és retorna a la pantalla taulell sense esborrar les dades.

4.3 Flux de pantalles.

El següent diagrama mostra el flux de pantalles que permet el programa.

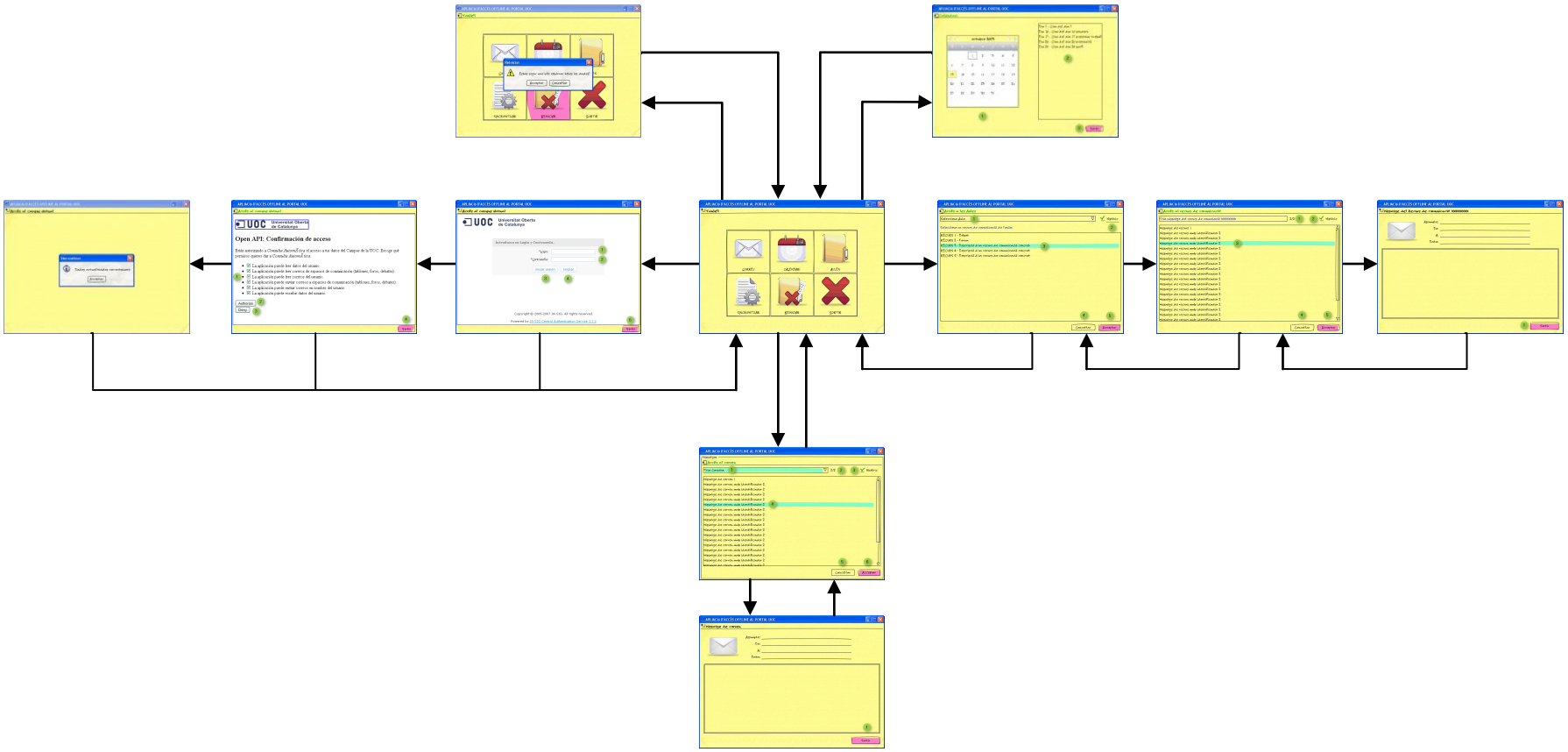

Figura 28 – Flux de pantalles.

4.4 Relació entre casos d'ús i pantalles.

Taula que relaciona els casos d'ús i les diferents pantalles.

Cas d'ús		Pantalles
Núm.	**Nom**	
1	*Login*	Benvinguda, Entrada contrasenya, Canvi contrasenya,
2	*Taulell*	Taulell
3	*Reiniciar*	Reiniciar
4	*Sincronitzar*	Sincronitzar
5	*Autenticació*	
6	*ConsultaCorreu*	Correu
7	*ConsultaMissatge*	Consulta Missatge
8	*ConsultaCalendari*	Consulta del calendari
9	*ConsultaAules*	Consulta de les aules Consulta dels missatges d'un recurs de comunicació
10	*ConsultaRecursComunicació*	Consulta d'un missatge d'un recurs de comunicació

Figura 29 – Taula de casos d'ús i pantalles.

4.5 Model conceptual de la BBDD.

En aquest apartat obtindrem l'estructura de la informació de la futura BD independent de la tecnologia final que acabarem utilitzant. Per fer-ho, utilitzarem el model *entity-relationship* o model *entitat-interrelació* **ER**.

4.5.1 Disseny conceptual. El model ER.

El model ER permet modelar les dades d'una forma força intuïtiva i entenedora. Ens permetrà reflectir en un model conceptual els requisits del món real d'interès per a la nostra aplicació.

Tenint en compte les necessitats de la nostra aplicació i fixant-nos en les dades i en els serveis proporcionats per l'OPEN API de la UOC, podrem fer el disseny conceptual de la nostra base de dades.

En una segona etapa i partint del disseny conceptual obtindrem el disseny lògic que s'adaptarà a la tecnologia que hàgim decidit utilitzar. En el nostre cas l'adaptarem al SGBD MySQL.

Finalment en una tercera etapa obtindrem el disseny físic tenint en compte aspectes d'implementació física que dependran del SGBD.
La següent figura mostra el diagrama ER que satisfà els requisits de la nostra aplicació:

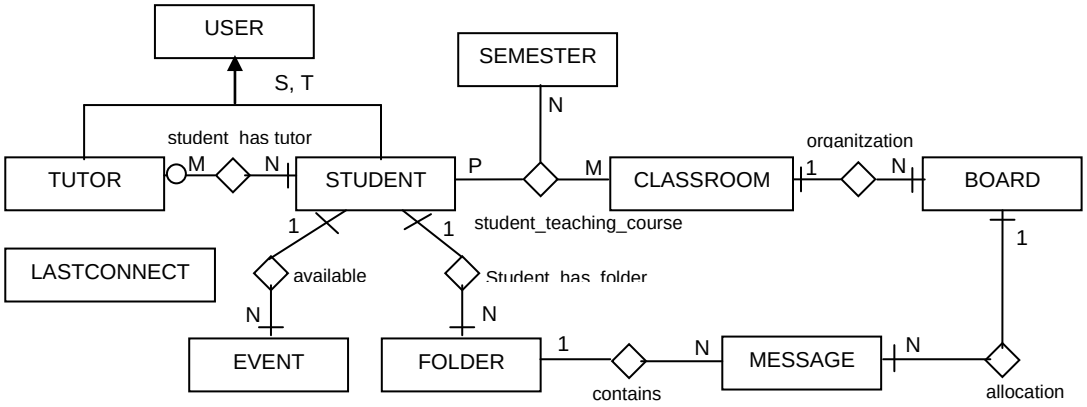

Figura 30 – Model ER.

L'especialització *User, Tutor, Student* és encavalcada (**S**) i total (**T**). Encavalcada, perquè he considerat que un usuari pot ser estudiant i un tutor alhora. Total, perquè tota ocurrència d'usuari ha de pertànyer a alguna de les entitats subclasse *tutor* o *student*.

També he considerat que un estudiant pot cursar la mateixa assignatura en diferents semestres. Aquest seria, per exemple, el cas en que un estudiant ha suspès una assignatura i el proper semestre s'hi torna a matricular.

Els atributs de les entitats que figuren al diagrama són els següents (les claus primàries s'han subratllat):

USER
    Id, username, name, number, fullName, photoUrl, language
STUDENT (subclasse d'user)
    Id
TUTOR (subclasse *d'user*)
    Id
SEMESTER
    year, name
MESSAGE
    Id, subject, date, snippet , status, color, from, to, cc, body
CLASSROOM
    Id, title, color, fathered, assignments, boards

BOARD
Id, title, subtype, domainId, code, totalMessages,
unreadMessages
FOLDER
Id, name, totalMessages, unreadMessages
EVENT
Id, url, summary, start, end
LASTCONNECT
Student_id, data
LOGIN
contrasenya

4.5.2 Disseny lògic. Del model ER al model Relacional.

En aquest apartat es tracta el disseny lògic de la base de dades relacional. Partirem del resultat del disseny conceptual expressat al model ER, i el transformarem en una estructura de dades del model relacional.

El resultat de la transformació del model ER al model relacional és la següent:

STUDENT(Id, username, name, number, fullName, photoUrl, language)
TUTOR (Id, username, name, number, fullName, photoUrl, language)
STUDENT_HAS_TUTOR(IdStudent,IdTutor)
On {idStudent} referencia a STUDENT
On {idTutor} referencia a TUTOR
EVENT(Id, idStudent, url, summary, start, end)
On {idStudent} referencia a STUDENT
FOLDER(Id, IdStudent, name, totalMessages, unreadMessages)
On {idStudent} referencia a STUDENT
SEMESTER(year, name)
CLASSROOM(Id, title, color, fathered, assignments, boards)
STUDENT_TEACHING_COURSE(IdStudent, yearSem, nameSem,
IdClass)
On {idStudent} referencia a STUDENT
On { yearSem, nameSem } referencia a SEMESTER
On { IdClass } referencia a CLASSROOM
BOARD(Id, IdClass, title, subtype, domainId, code, totalMessages,
unreadMessages
On { IdClass } referencia a CLASSROOM

MESSAGE(Id, IdBoard, IdFolder, subject, date, snippet , status, color,
from, to, cc, body
On { IdBoard } referencia a BOARD
On { IdFolder } referencia a FOLDER
LASTCONNECT(User_id, data)
LOGIN (contrasenya)

Comentaris sobre el procés de transformació:

- La transformació de l'especialització fa que USER desaparegui i que els atributs es copiïn a TUTOR i STUDENT. Una altra opció consistiria en mantenir la classe USER i posar el seu *id* a les taules TUTOR i STUDENT.
- La interrelació ternària (M:N:P) fa aparèixer una nova relació anomenada STUDENT_TEACHING_COURSE.
- La interrelació binària M:N entre TUTOR i STUDEN fa aparèixer una nova relació STUDENT_HAS_TUTOR.
- La transformació de la resta d'interrelacions ha consistit en afegir l'atribut d'una de les relacions implicades a l'altra també implicada segons el sentit de la navegació.
- La taula *LastConnect* permet emmagatzemar l'usuari i la data que ha efectuat la darrera actualització amb el campus UOC.

4.5.3 Disseny físic.

En aquest apartat es tracta de realitzar el disseny físic de la BD tenint en compte un SGBD concret. En el nostre cas, el SBGD és **MySql** en la seva versió 5.5.8.

Per generar el següent diagrama EER s'ha utilitzat l'eina *MySQL Workbench community* en la seva versió 6.0.7 per al sistema operatiu Windows de 32 bits. Podem descarregar l'eina des del següent enllaç:

http://dev.mysql.com/downloads/tools/workbench/

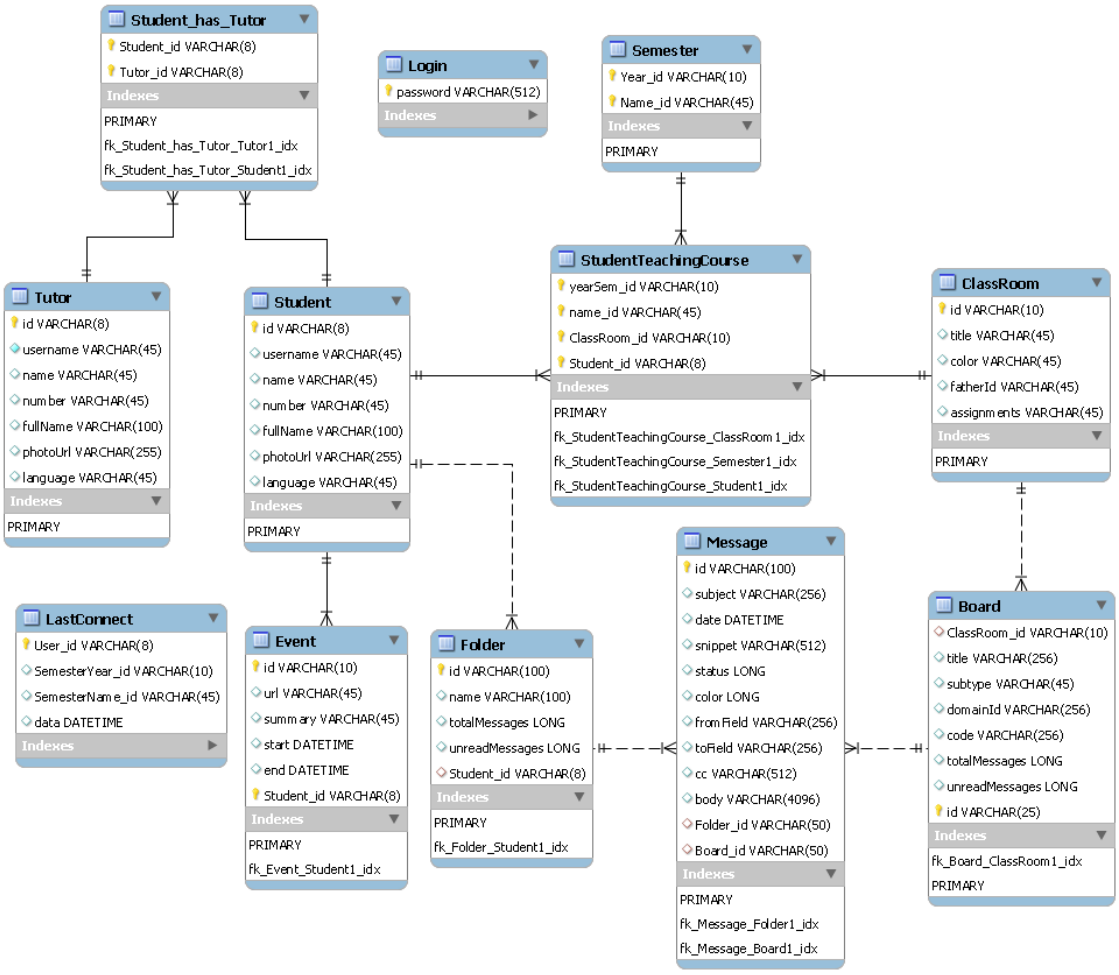

Figura 31 – Disseny físic de la BD.

En la normalització de la base de dades s'ha optat per suprimit l'herència i copiar tots els atributs a cada una de les classes hereves, *tutor* i *student*. A més han aparegut dues noves taules. Una taula és "*Student_has_Tutor*" fruit de la relació molts a molts entre Tutor i Student i una altra taula és "*StudentTeachingCourse*" fruit de la relació ternària entre les taules *Semester, Student* i *ClassRoom*.

La taula *Login* permet emmagatzemar la contrasenya d'entrada al programa. Només té un camp *password* que permet emmagatzemar la contrasenya xifrada mitjançant un algorisme *Hash* **SHA-2** de 512 bytes.

La taula *LastConnect* emmagatzema informació sobre l'usuari i la data de la darrera connexió. Tota la informació que mostra el programa fa referència a les dades de l'usuari que ha executat l'opció de sincronització per darrera vegada.

He afegit les claus foranes necessàries segons les cardinalitats i les relacions establertes al model relacional.

Nota: No he disposat ni dels tipus ni de la mida dels camps de les taules originals de la UOC. Per aquest motiu no puc assegurar que la seva mida sigui vàlida al 100%. Només puc assegurar que funciona perfectament amb l'entorn actual de proves.

5.Implementació.

5.1 Decisions tecnològiques.

Per desenvolupar el projecte hem d'analitzar els requisits tecnològics necessaris que ens permetin accedir a les dades de la UOC, tractar-les i representar-les a l'usuari amb un entorn gràfic. Ens cal doncs analitzar com resoldre tecnològicament les següents qüestions:

- Accedir de forma segura a *l'API* de la UOC.
- Fer peticions *HTTP*.
- Tractar dades en format *JSON*.
- Utilitzar components i recursos gràfics (*Swing*).
- Gestionar dades amb una *BBDD* relacional.

Tot seguit veurem quins recursos tecnològics donen solució als diferents requisits esmentats.

5.1.1 La llibreria *DJ Project.*

L'aplicació ha de permetre accedir de forma segura a la UOC. L'estratègia utilitzada per resoldre l'autenticació segura de l'usuari al campus UOC, consistirà en l'ús de la llibreria *DJ Project*[5]. Aquesta llibreria, nativa en Swing, permet d'integrar un navegador web encastat a la nostra aplicació. Aquesta estratègia em permetrà utilitzar els formularis d'autenticació de la UOC. La web de descarrega és:

https://github.com/Chrriis/DJ-Native-Swing

5.1.2 La llibreria Apache Oitu OAuth 2.0

També s'utilitzaran les llibreries client *Apache Oitu OAuth 2.0* que podem descarregar del següent enllaç:

http://oltu.apache.org/download.html

Aquestes llibreries, en arquitectures que utilitzen el protocol d'autenticació *OAuth* permeten obtenir, mitjançant codi, el *token* o codi d'autorització que necessita l'aplicació per accedir als serveis de l'API. El següent diagrama d'activitats mostra com aconseguir el *token* al servidor de la UOC:

[5] Llibreries de components desenvolupades integrament utilitzant les llibreries *Swing*.

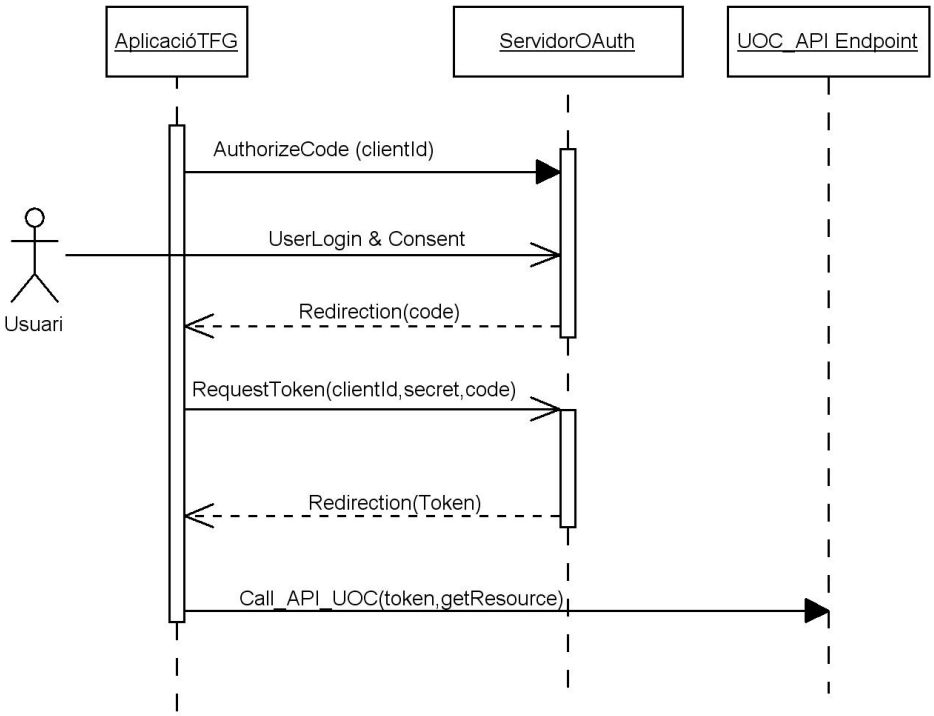

Figura 32 – Autenticació OAuth.

Les especificacions *OAuth 2.0 Specification* les podem trobar a l'RFC 6749.

5.1.3 La llibreria HttpClient d'Apache Software Foundation.

Un cop autenticats i amb un *token* vàlid per accedir a l'API, ens caldrà un client que ens permeti fer peticions *HTTP_GET* per tal d'obtenir les dades del campus. S'utilitzaran les llibreries de codi lliure *HttpClient 4.3.1* també d'*Apache Software Foundation* que podem descarregar del següent enllaç:

http://hc.apache.org/downloads.cgi

5.1.4 La llibreria google-gson.

Les consultes a l'API és faran en format *JSON*. Aquest format és un subconjunt de la notació literal dels objectes *JavaScript* que no necessiten l'ús *d'XML*[6]. La simplicitat de *JSON* ha permès la generalització del seu ús, especialment com alternativa a *XML* en *AJAX*[7].

[6] ***eXtensible Markup Language.***
[7] ***Asynchronous JavaScript And XML***

Per tant necessitem alguna llibreria que ens permeti convertir cadenes *JSON*[8] en objectes de tipus Java i a l'inrevés.

Aquest cop la solució més simple ha estat incorporar la llibreria *google-gson.* Aquesta llibreria em permet fer exactament el que cercàvem fins i tot, si no disposéssim del codi font de les classes *Java*. Les llibreries es poden descarregar des del següent enllaç:

http://code.google.com/p/google-gson/

5.1.5 La GUI[9] amb Swing.

Com s'ha comentat anteriorment per desenvolupar la interfície gràfica d'usuari s'utilitzaran les llibreries *Swing*. Aquestes llibreries estan incorporades amb el *JDK*[10] de Java. Podem trobar la documentació al següent enllaç:

http://docs.oracle.com/javase/tutorial/uiswing/start/index.html

5.1.6 El SGBD amb MySQL.

Per tal d'emmagatzemar les dades de forma local s'utilitzarà un sistema gestor de bases de dades. Ens interessa una BBDD lliure, de fàcil ús i d'eficiència contrastada. Així doncs, penso que la solució més adient és utilitzar *MySQL*.

5.1.7 L'IDE amb Eclipse.

L'entorn **IDE**[11] utilitzat per al desenvolupament del projecte serà *l'Eclipse* que ens podem descarregar des del següent enllaç.

http://www.eclipse.org/downloads/

Aquest IDE és un del més utilitzat en entorns de programació Java juntament amb *JavaBeans*.

[8] *JavaScript Object Notation.*
[9] ***G**raphical **U**ser **I**nterfaces*
[10] Java Development Kit
[11] ***I**ntegrated **D**evelopment **E**nvironment.*

5.1.8 SHA-2 per a Java.

L'algorisme de xifratge SHA-2, en espera del seu successor SHA-3, és actualment el més segur. S'espera que el nou algorisme SHA-3 anomenat "*Keccak*" aparegui durant la segona meitat del 2014.

Utilitzarem la llibreria *"java.security.MessageDigest*" per xifrar la contrasenya d'entrada al programa.

5.2 La capa de presentació.

Passem a veure la implementació de la capa de presentació.

En aquesta capa hi haurà una classe façana controladora que canalitzarà les peticions a la seva capa adjacent, i com ja havíem comentat, a totes les pantalles aplicarem el patró MVC. Podem observar-ho en la següent figura:

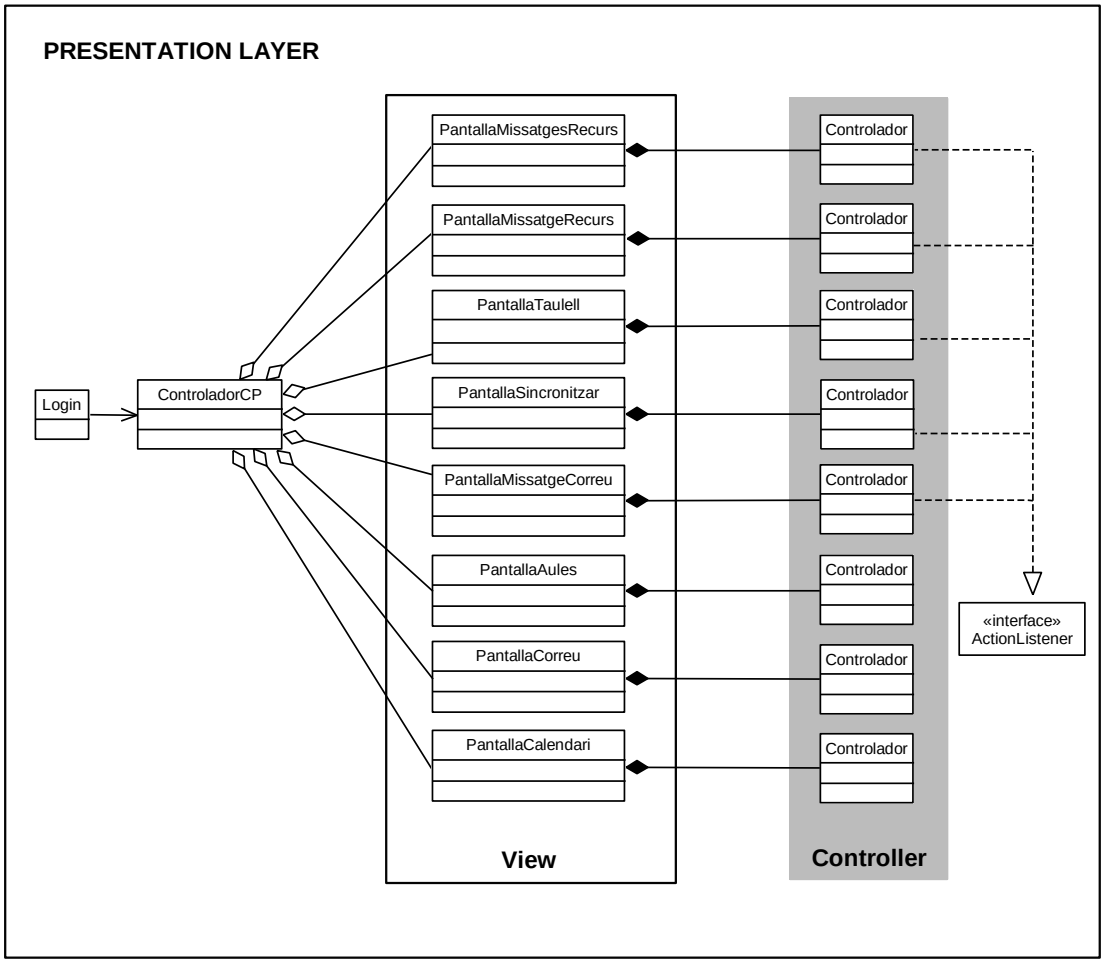

Figura 33 – Capa de presentació.

El model estaria format per la resta de capes.

Totes les classes pantalla tenen una classe niuada que fa de controladora, atenent les peticions de la interfície gràfica. Per exemple, per la classe pantalla *Taulell* tindríem:

```
public class PantallaTaulell extends JFrame {
        ...
        class Controlador implements ActionListener{
                ...
        }
}
```

Per altra banda la classe controladora de la capa de presentació "*ControladorCP*" tracta les peticions de la capa de presentació a la capa adjacent, la capa de *Domini/Regles de Negoci*.

5.3 La capa de Domini/Regles de Negoci.

En aquesta capa hi haurà el model de dades i una altra classe façana *ControladorBD* que implementarà i centralitzarà les peticions a la Base de dades. La classe façana utilitzarà una altra classe *GestorBD* per fer les connexions a la base de dades, tal com mostra la següent figura:

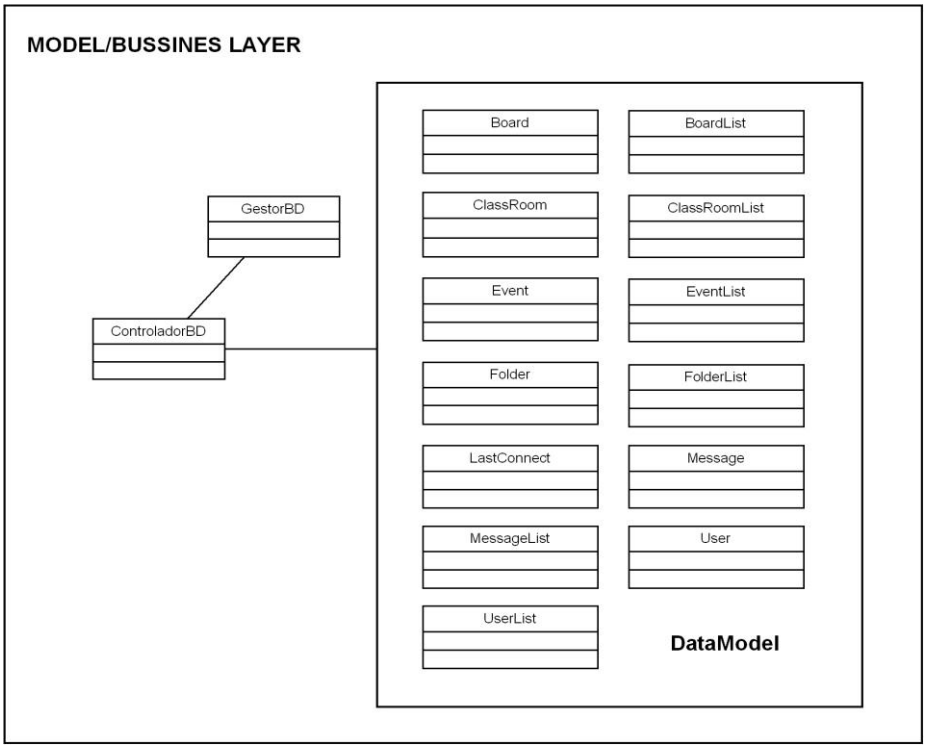

Figura 34 – Capa de *Domain/Bussines.*

5.4 La Capa de Dades.

En aquesta capa hi ha la base de dades "**CampusDB**" que ens permet salvar localment les dades consultades al campus. La següent figura ho mostra:

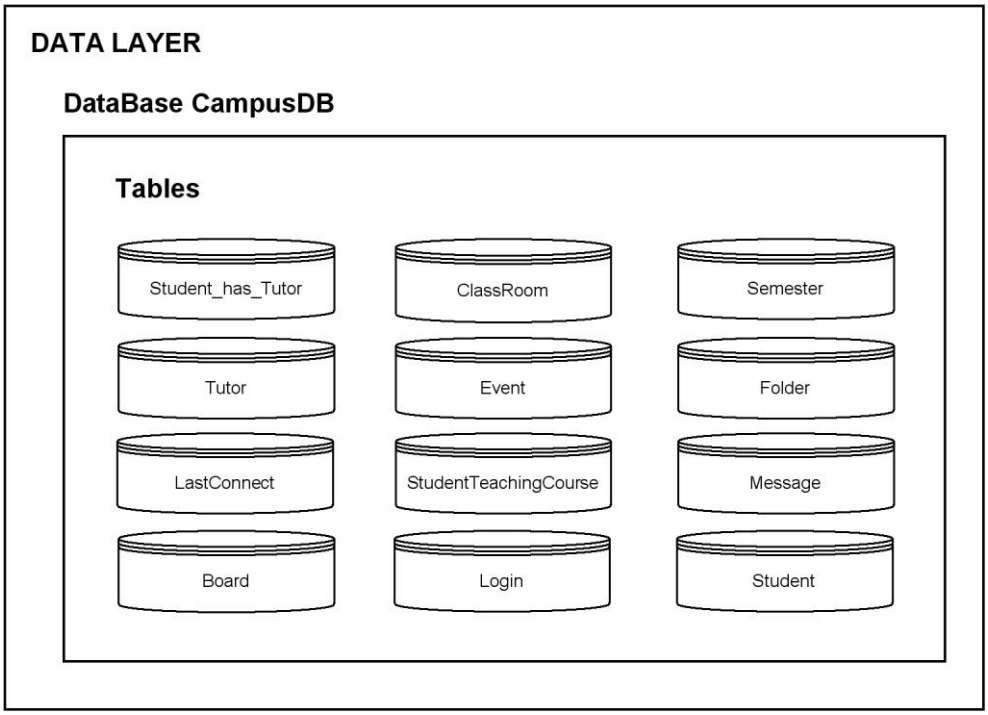

Figura 35 – Data Layer.

S'annexa un fitxer "*scriptBD.sql*" amb les consultes SQL que permeten afegir la base de dades a MySQL. Al final del document s'annexa una còpia del contingut d'aquest fitxer.

Les consultes comproven si ja existeix la base de dades, cas en que l'esborra. Després la crea de nou juntament amb la resta de taules. També es crea l'usuari "*userCampus*" amb la contrasenya "*tfguoc2013"* amb tots els drets sobre la base de dades. Podem canviar aquestes dades que es troben al fitxer *Constants.java* del projecte.

Inicialment les consultes SQL configuren l'accés local (*localhost*) a la base de dades. Si volem permetre la connexió a qualsevol de les interfícies del nostre equip només ens caldria treure els comentaris a les següents sentències que hi ha al final del fitxer:

```
/*
GRANT ALL ON CampusDB.* TO 'userCampus'@'%';
GRANT ALL ON TABLE CampusDB.* TO 'userCampus'@'%';
*/
```

En el següent esquema podem observar una vista completa de la implementació del model arquitectònic:

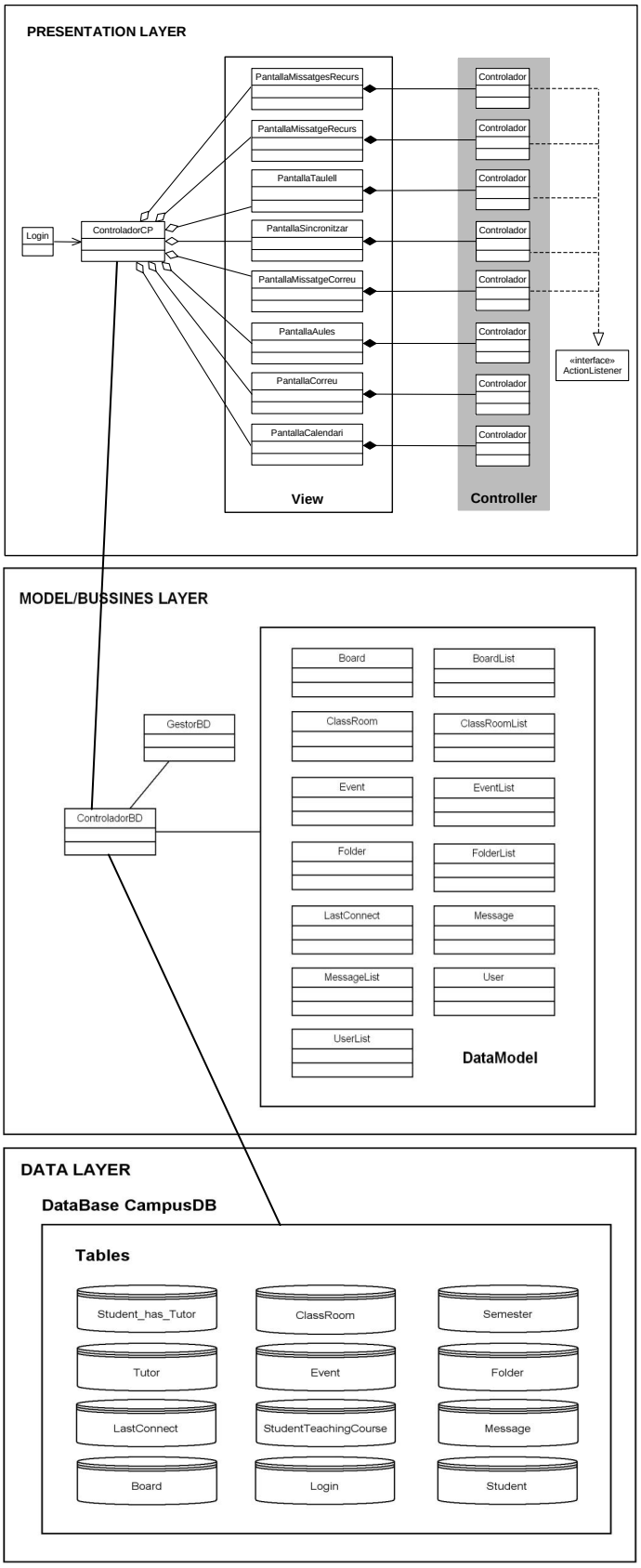

Figura 36 – Implementació del model arquitectònic.

5.5 Obtenir del *token*.

La classe *PantallaSincronitzar* és l'encarregada d'autenticar l'estudiant a la UOC i obtenir el *token* d'accés. El constructor d'aquesta classe comprova si ja tenim el *token*. En aquest cas només actualitza les dades. Altrament s'autentica, demana el *token* i actualitza les dades. Aquest és el codi:

```
if(controladorCp.getOauthToken().equals(""))   autenticarse ();
else {
        lblMissConnectant.setVisible(false);
        tascaPesadaSincronitzaDadesUOC();
}
```

El mètode autenticar-se al seu torn també crida al mètode:
tascaPesadaSincronitzaDadesUOC();

Per tant, analitzarem el mètode autenticar-se.

Per aconseguir l'autenticació amb el protocol d'autorització *OAuth* i obtenir el codi, utilitzem les llibreries *Apache Oltu libreries*. Analitzem el codi:

```
OAuthClientRequest request=null;
request = OAuthClientRequest
   .authorizationLocation(Constants.SERVER+Constants.AUTHORIZE_URI)
   .setClientId(Constants.CLIENT)
   .setRedirectURI(web1) //adreça de retorn
   .buildQueryMessage();
```

On els paràmetres són (disponibles a la classe *Constants.java*):

```
public static String SERVER="http://oslo.uoc.es:8080/";
public static String AUTHORIZE_URI ="webapps/uocapi/oauth/authorize";
public static String REDIRECT_URI="html/retorn.html";
Constants.CLIENT és la clau d'accés a l'API facilitada per la UOC.
public String web1 =
WebServer.getDefaultWebServer().getClassPathResourceURL(getClass().getName(),
                                                    Constants.REDIRECT_URI);
```

El mètode *getClassPathResourceURL* serveix per localitzar la referència o adreça interna de l'arrel del navegador web encastat. D'aquesta forma podem fer referència a l'adreça de la pàgina web *retorn.html* ubicada dins del nostre projecte. Aquesta és la pàgina on el servidor retornarà el resultat de la petició del *codi*. Aquesta adreça origen, per un tema de

seguretat, no pot canviar la seva ubicació durant l'execució d'obtenció del *token*.

Finalment es concatena l'adreça web amb el paràmetre *code* i procedim a la seva petició al navegador encastat:

```
webuoc=request.getLocationUri().toString()+"&response_type=code";
webBrowser.navigate(webuoc);
```

El mètode:

```
public void locationChanged(WebBrowserNavigationEvent arg0)
```

Permet analitzar la resposta del servidor amb el següent codi:

```
if (adreca.contains("code=")){
      this.setCode(adreca.substring(adreca.indexOf("=")+1));
      tascaPesadaSincronitzaDadesUOC();
}
```

La resposta té èxit si duu a l'adreça de retorn el paràmetre "*code*=". A continuació salvem el codi a l'atribut de classe i executem la tasca de sincronització de dades *tascaPesadaSincronitzaDadesUOC.*

En aquesta tasca s'obté el *token* a partir del *code* obtingut. El codi és el següent:

```
      if(controladorCp.getOauthToken().equals("")) {
            oauthToken=obtenirCLAU();//demana nou token
            controladorCp.setOauthToken(oauthToken);//salva el token
      }else oauthToken=controladorCp.getOauthToken();
```

Si no disposem del *token* fem la petició al mètode *obtenirCLAU().* La clau obtinguda l'assignem a la variable *oautToken* i també la salvem a l'atribut de classe. Si ja disposem del *token,* simplement, l'assignem a la variable *oauthToken* per tenir-la disponible.

El codi del mètode *obtenirCLAU* és:

```
request = OAuthClientRequest
.tokenLocation(Constants.SERVER+Constants.ACCESS_TOKEN_URI)
.setGrantType(GrantType.AUTHORIZATION_CODE)
.setClientId(Constants.CLIENT).setClientSecret(Constants.SECRET)
.setRedirectURI(web1).setCode(code)// mateixa URI inicial
.buildBodyMessage();

OAuthClient oAuthClient = new OAuthClient(new URLConnectionClient());
OAuthJSONAccessTokenResponse response= oAuthClient.accessToken(request);
oauthToken= response.getAccessToken();
```

On els paràmetres són:

public static String **SERVER**="http://oslo.uoc.es:8080/";
public static String **ACCESS_TOKEN_URI**="webapps/uocapi/oauth/token";
GrantType.**AUTHORIZATION_CODE és una constant de la classe GrantType que informa del tipus de petició: autorització de codi.**
Constants.**CLIENT** és la clau d'accés a l'API facilitada per la UOC.
Constants.**SECRET** és el secret d'accés a l'API facilitada per la UOC.
Web1 és l'adreça de retorn trobada anteriorment.
code és el codi obtingut en la petició anterior.

Finalment es crea un objecte de tipus *OAutClient* que embolcalla la petició *request* i s'obté un objecte *OAuthJSONAccess* per rebre la resposta de tipus *JSON*.

Un cop disposem del *token* d'autorització, la resta de codi de la classe *tascaPesadaSincronitzaDadesUOC* s'encarrega de fer les consultes i emmagatzemar les respostes a la base de dades local.

5.6 La classe SwingWorker.

El mètode *tascaPesadaSincronitzaDadesUOC* s'executa a l'EDT[12] o fil d'execució de successos. Aquest és el fil on s'executen els *listeners* dels objectes Swing. Cal tenir cura de no executar tasques pesades en aquest fil o podríem "*congelar*" la part gràfica. Per aquest motiu aquest mètode crea un nou fil d'execució mitjançant la classe *SwingWorker*. Aquesta classe Java facilita l'execució i sincronisme de tasques pesades en un fil diferent de l'EDT.

També disposa de mecanismes de publicació de resultats parcials o d'un resultat final del fil pesat al fil EDT.

[12] **E**vent **D**ispach **T**hread.

La tasca pesada l'hem d'executar al mètode *doInBackground* i el resultat o possible retorn l'obtindrem amb el mètode *done*.

En el mètode *doInBackground* executem successivament totes les consultes requerides al programa. Per exemple, el mètode de sincronització de l'usuari seria:

```
sincronitzaUsuari(oauthToken)
```

El codi d'aquest mètode és:

```
private User sincronitzaUsuari(String oauthToken){
    String StringJSONResource =UOCjsonTools.getJSONStringResource(
    UOCjsonTools.URI_ROOT+
    UOCjsonTools.URI_USER,
    oauthToken);

    User usuari= (User)
    UOCjsonTools.JSONToDataModel(StringJSONResource, User.class);

    if(!controladorCp.altaAlumne(usuari)) return null;
    return usuari;
}
```

On els paràmetres són (podem trobar aquestes constants definides a la classe *UOCjsonTools*):

URI_ROOT=Constants.SERVER+"webapps/uocapi/api/v1/"
URI_USER="user";
oauthToken el *token* d'autorització obtingut.

Amb el mètode *getJSONStringResource* obtenim el resultat de la consulta JSON amb format cadena.

Amb el mètode J*SONToDataModel* passem de la cadena de consulta obtinguda a la classe corresponent Java, en aquest cas *User.class*.

Finalment fem una petició al mètode *altaAlumne* que enregistra les dades a la base de dades local i retorna l'objecte de la classe **User** (*usuari)* que representa l'usuari consultat.

Totes les peticions de consulta tenen un format similar.

També hi ha un objecte de la classe *JProgressBar* que mostra el progrés d'actualització de les diferents consultes.

5.7 Les consultes a l'API i les classes genèriques.

Tota la gestió de consultes a l'API de la UOC es concentra a la classe "*PantallaSincronitzat*". Aquesta classe és la més important i fa les següents accions:

- Es connecta amb l'API de la UOC,
- Executa consultes de tipus JSON,
- Converteix les dades JSON a les classes Java corresponents que configuren el model de dades,
- Finalment salva les dades a la base de dades local.

La classe "*UOCjsonTools.java*" disposa d'un conjunt de constants amb l'URI corresponent per accedir als diferents recursos de l'API. Per fer les consultes s'utilitza la llibreria *Apache HTTP Client* que permet fer consultes *HttpGet*. En aquesta classe es controlen els possibles errors d'accés a l'API.

El mètode genèric *JSONToDataModel* permet passar del model de dades JSON a classes Java. El codi és el següent:

```
public static <T> Object JSONToDataModel(String jsonString, Class<T> classe)
{
   Gson gson=null;
  if (classe==Event.class || classe==EventList.class ){
            gson=  new GsonBuilder().setDateFormat("yyyy-MM-dd'T'HH:mm:ss").create();
         }else{
     if (classe==MessageList.class ||classe==Message.class ){
        gson=  new GsonBuilder().setDateFormat("EEE MMM dd hh:mm:ss z yyyy").create();
     }else{
        gson=  new GsonBuilder().create();
     }
  }
  T obj=gson.fromJson(jsonString, classe);
  return obj;
}
```

La classe està parametritzada amb un *template* **<T>** que permet fer la classe genèrica. D'aquesta forma podem convertir qualsevol dada de tipus JSON en la seva classe equivalent Java.

En temps d'execució es decidirà el tipus del *template* <T>. A més, l'API facilita dades de tipus *data* amb dos formats diferents. Un tipus de *data* es troba a les cites del calendari i l'altra als missatges. Per aquest motiu hi ha un filtre segons el tipus de classe. Segons si es tracta d'una o altra classe, s'estableix un format concret de data.

El pas d'un model a l'altre l'efectua el servei *gson.fromJson(json,classe).*

El mètode *getJSONStringResource(String URI, String token)* és l'encarregat de fer les consultes *HttpGet* de tipus *JSON* a *l'URI* corresponent. El codi més rellevant seria:

```
HttpGet httpGet = new HttpGet(URI+"?access_token="+token);
httpGet.setHeader("content-type", "application/json");
HttpResponse response = httpClient.execute(httpGet);
```

L'*URI* es rep per paràmetre juntament amb el *token*. El codi concatena l'*URI* amb el paràmetre *token* per fer la consulta *HttpGet*. Després afegeix una capçalera indicant que el contingut serà de tipus *json* i finalment executa la consulta.

Si la consulta té èxit retorna una cadena que representa les dades *json*. La següent línia de codi crea la cadena:

```
JSONString = EntityUtils.toString(response.getEntity());
```

Posteriorment es faria una crida al mètode *JSONToDataModel* amb la cadena i la classe Java equivalent com a paràmetres.

5.8 El *debugger*.

La classe *PantallaSincronitzar* disposa d'un petit depurador. La variable *debug* controla el comportament. Si té valor *true* permet depurar els resultats de les consultes a la UOC mitjançant la consola o terminal. Si té valor *false* no mostra els resultats per consola.

Inicialment l'opció està activada i permet observar a la consola les dades consultades en el mateix moment de fer la consulta. Pot resultar força útil per comprovar errors i consultar les dades obtingudes pas a pas.

Podem desactivar aquesta opció assignant valor *false* a la variable *debug.*

5.9 Control d'errors i robustesa.

Tot seguit analitzem les diferents situacions que poden produir errors i que cal controlar. També tindrem en compte determinats filtres d'entrada per fer el codi robust.

Escenari: Primer cop que s'executa del programa.

Figura 37 – Error en la base de dades

Si no té connexió amb la base de dades i no pot enregistrar la contrasenya del programa, no té sentit seguir i surt del programa.

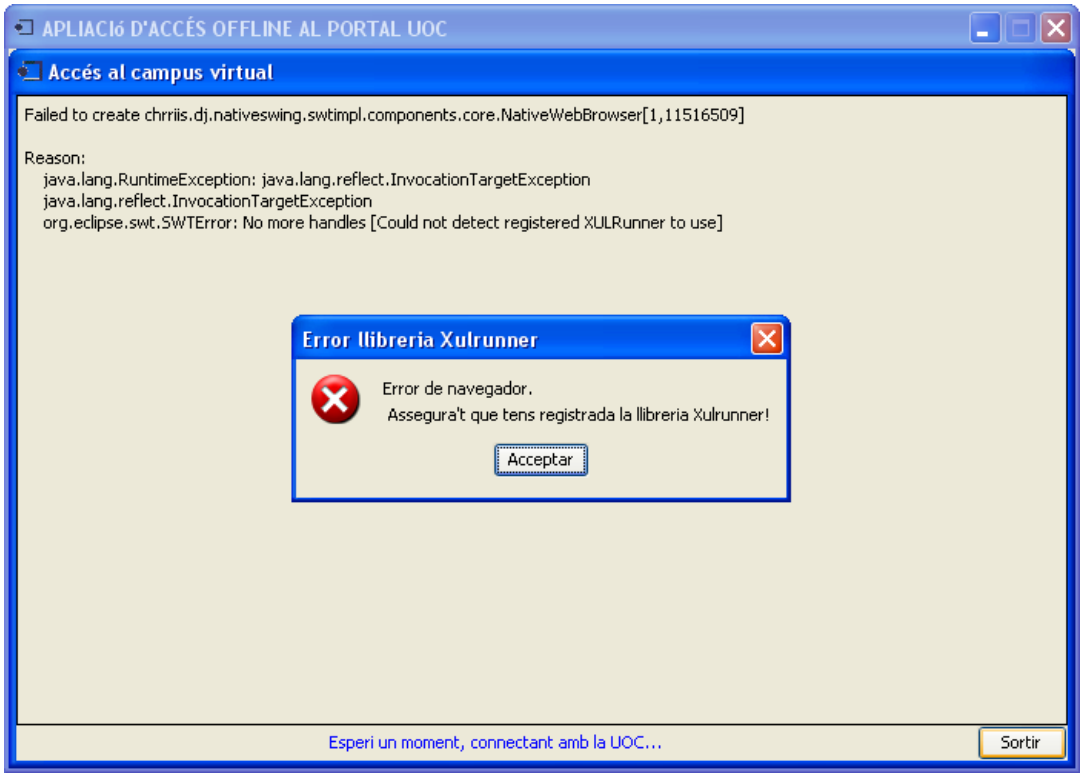

Figura 38 – Error en la llibreria XulRunner

El programa no detecta *XulRunner* registrat al sistema.

És necessari tenir instal·lades correctament al sistema les llibreries *XulRunner*. Aquestes llibreries permeten executar el motor de *Mozilla* al navegador i fer la connexió amb la UOC via Internet.

Escenari: S'està executant la sincronització de dades.

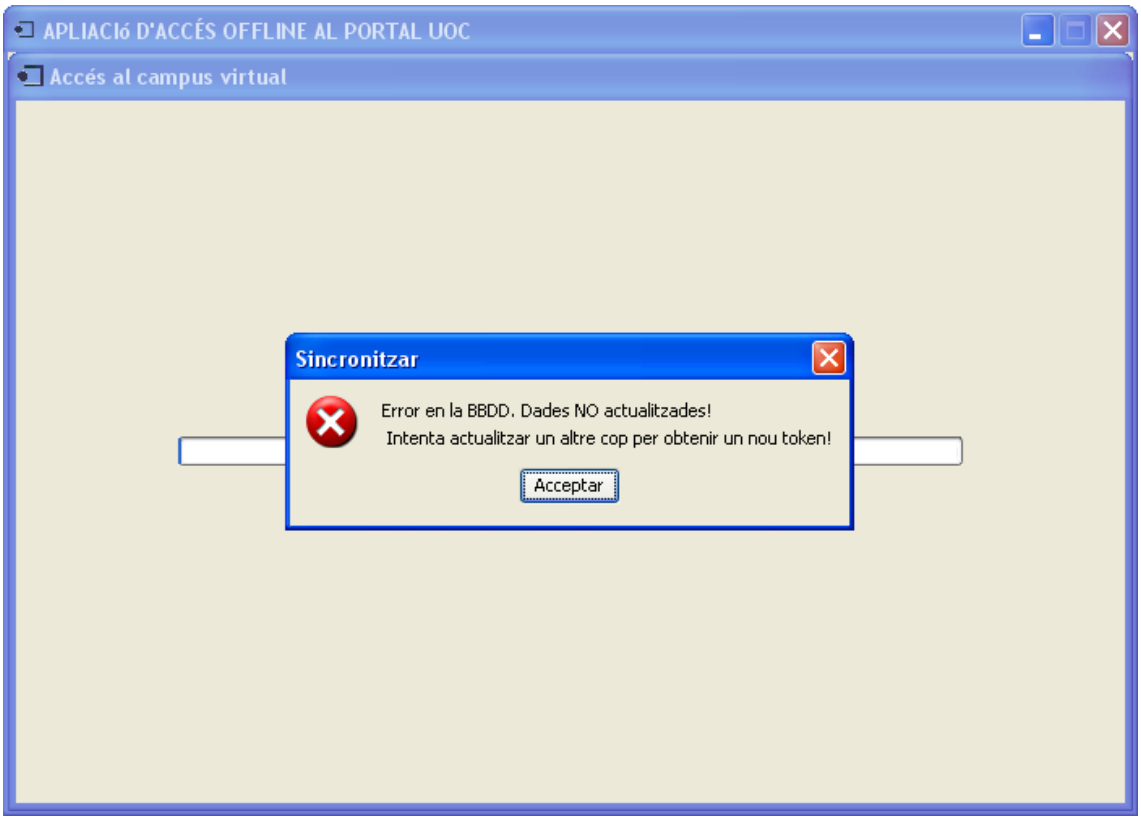

Figura 39 – Error de sincronització.

Error en la sincronització de dades. La connexió no ha tingut èxit, per tant, les dades no s'han actualitzat. Aquest missatge també recull l'error en el *token*. Una solució seria tornar a executar l'opció *sincronitzar*.

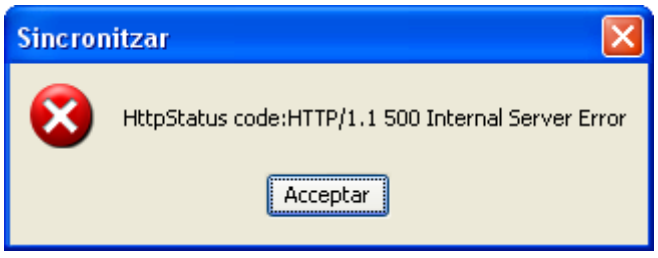

Figura 40 – Error intern del Servidor UOC.

La solució de l'error intern del servidor consisteix en tornar a executar l'opció *sincronitzar*.

Figura 41 – Connexió rebutjada pel servidor *oslo.uoc.es*

No hem pogut connectar amb el servidor, la connexió ha estat rebutjada. La solució consisteix en tornar a executar l'opció *sincronitzar*. Si segueix l'error cal comprovar la connexió de l'equip a Internet.

Escenari: És el primer cop que s'entra al programa i estem introduint la contrasenya que en protegirà l'accés.

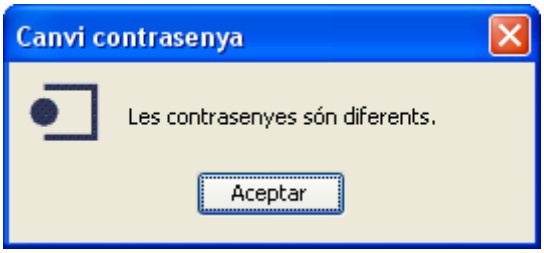

Figura 42 – Primera contrasenya. Error contrasenyes diferents.

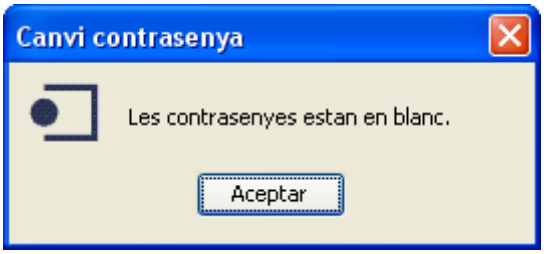

Figura 43 - Primera contrasenya. Error contrasenyes en blanc.

Escenari: Introducció de la contrasenya per entrar al programa.

Figura 44 – Contrasenya en blanc.

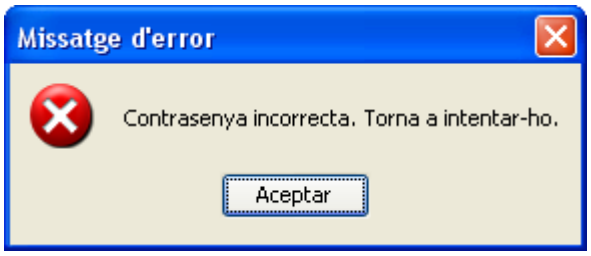

Figura 45 – Contrasenya incorrecta.

Escenari: Volem canviar la contrasenya.

Per canviar la contrasenya cal introduir la contrasenya actual i després demana dues vegades la nova contrasenya.

Figura 46 – Canvi de contrasenya. Contrasenya en blanc.

Aquest error fa referència a la contrasenya actual.

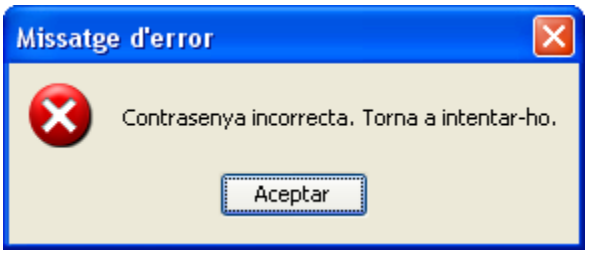

Figura 47 – Canvi de contrasenya. Contrasenya incorrecta.

Aquest error fa referència a la contrasenya actual.

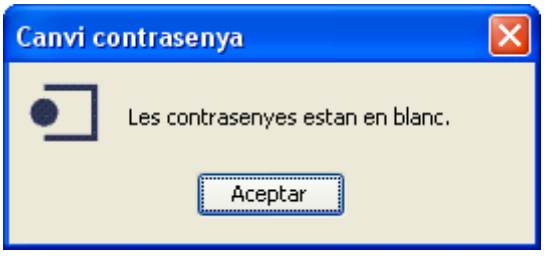

Figura 48 – Canvi de contrasenya. Avis de contrasenyes en blanc.

En els següents missatges d'avís, se suposa que la contrasenya actual s'ha introduït correctament, però la nova contrasenya està en blanc en el primer cas; o bé la nova contrasenya, que s'ha introduït dos cops, no coincideix.

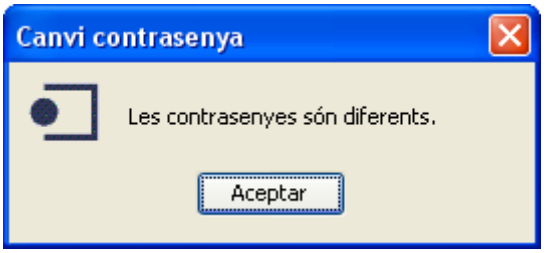

Figura 49 – Canvi de contrasenya. Contrasenyes diferents.

Escenari: Volem consultar els missatges de les carpetes de correu.

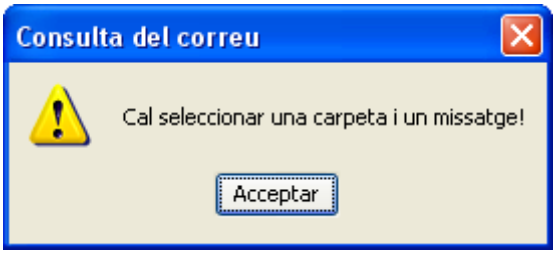

Figura 50 – Avis consulta correu.

Ens informa que per visualitzar un missatge primer cal seleccionar una carpeta i triar el missatge que volem visualitzar.

Escenari: Volem consultar les aules, els seus recursos i els missatges dels recursos.

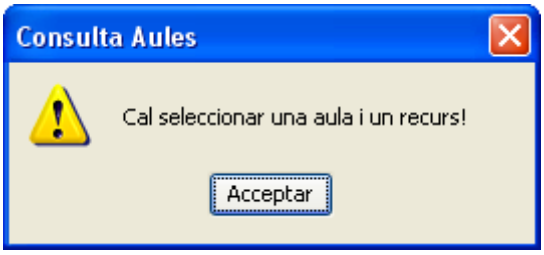

Figura 51 – Avís selecció d'aula i recurs.

Figura 52 – Avis selecció missatge.

5.10 Joc de proves.

Després d'una sincronització, les dades retornades per consola són les següents:

```
FoldersList [folders=[
Folder [id=1386114815221_1386114815221_1, name=Nom de la carpeta amb
id 1386114815221_1386114815221_1, totalMessages=3, unreadMessages=3],
Folder [id=1386114815221_1386114815221_2, name=Nom de la carpeta amb
id 1386114815221_1386114815221_2, totalMessages=3, unreadMessages=3],
Folder [id=1386114815221_1386114815221_3, name=Nom de la carpeta amb
id 1386114815221_1386114815221_3, totalMessages=3, unreadMessages=3],
Folder [id=1386114815221_1386114815221_4, name=Nom de la carpeta amb
id 1386114815221_1386114815221_4, totalMessages=3, unreadMessages=3]]]
```

La següent pantalla mostra les carpetes de correu:

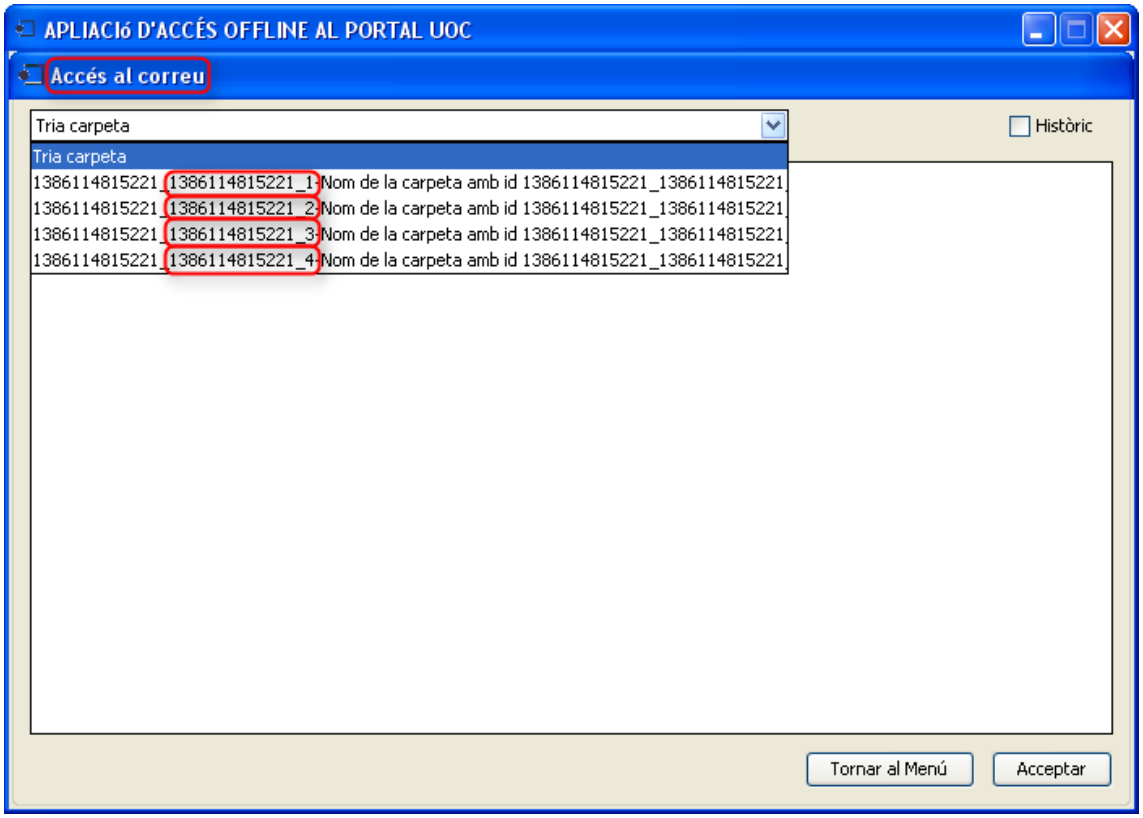

Figura 53 – Joc de proves. Consulta de correu 1.

Els missatges llegits per consola de la carpeta amb id 1386114815221_2 són:

```
Read: MessageList [messages=[
Message [body=Cos del missatge del correu amb id
1386114815221_1386114815221_21386114815922_1, cc=Destinataris en cï¿½pia del
missatge amb id 1386114815221_1386114815221_21386114815922_1, color=0,
date=Wed Dec 04 00:53:35 CET 2013, from=Remitent del missatge amb id
1386114815221_1386114815221_21386114815922_1,
id=1386114815221_1386114815221_21386114815922_1, snippet=Primers caracters del
missatge amb id 1386114815221_1386114815221_21386114815922_1, status=0,
subject=Tema del missatge amb id 1386114815221_1386114815221_21386114815922_1,
to=Destinataris del missatge amb id
1386114815221_1386114815221_21386114815922_1],
Message [body=Cos del missatge del correu amb id
1386114815221_1386114815221_21386114815922_2, cc=Destinataris en cï¿½pia del
missatge amb id 1386114815221_1386114815221_21386114815922_2, color=0,
date=Wed Dec 04 00:53:35 CET 2013, from=Remitent del missatge amb id
1386114815221_1386114815221_21386114815922_2,
id=1386114815221_1386114815221_21386114815922_2, snippet=Primers caracters del
missatge amb id 1386114815221_1386114815221_21386114815922_2, status=0,
subject=Tema del missatge amb id 1386114815221_1386114815221_21386114815922_2,
to=Destinataris del missatge amb id
1386114815221_1386114815221_21386114815922_2],
Message [body=Cos del missatge del correu amb id
1386114815221_1386114815221_21386114815922_3, cc=Destinataris en cï¿½pia del
missatge amb id 1386114815221_1386114815221_21386114815922_3, color=0,
date=Wed Dec 04 00:53:35 CET 2013, from=Remitent del missatge amb id
1386114815221_1386114815221_21386114815922_3,
id=1386114815221_1386114815221_21386114815922_3, snippet=Primers caracters del
missatge amb id 1386114815221_1386114815221_21386114815922_3, status=0,
subject=Tema del missatge amb id 1386114815221_1386114815221_21386114815922_3,
to=Destinataris del missatge amb id
1386114815221_1386114815221_21386114815922_3]]]
```

La pantalla corresponent és:

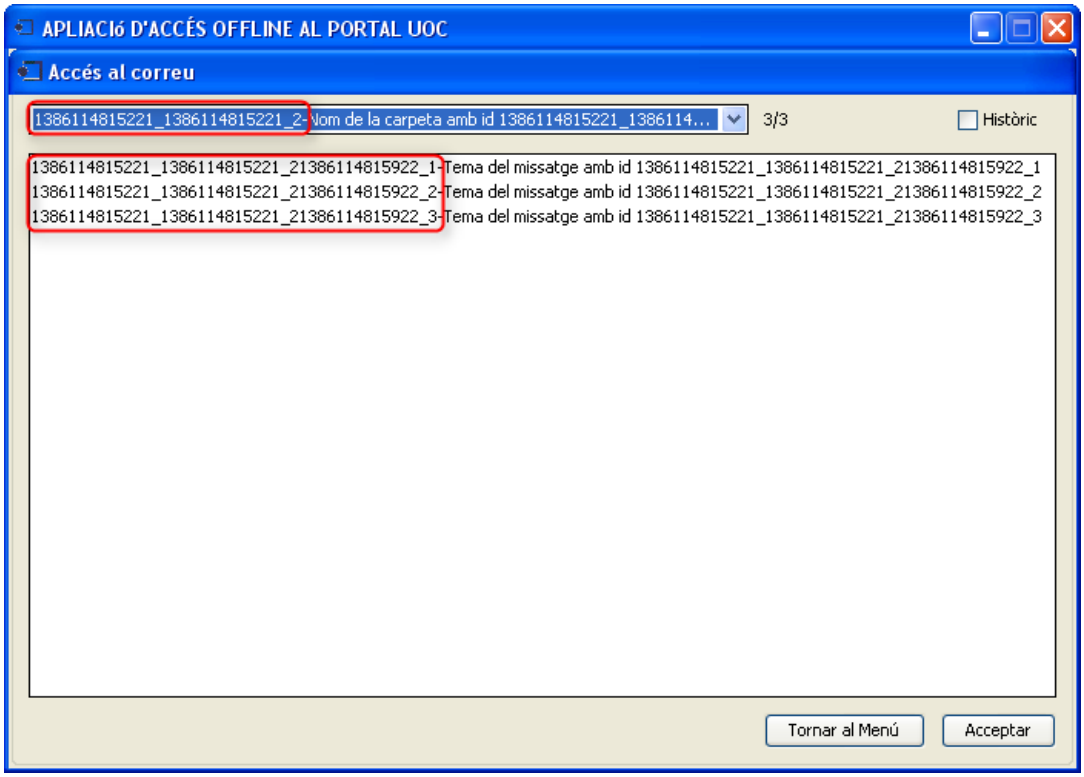

Figura 54 – Joc de proves. Consulta de correu 2.

Les dades llegides per consola del missatge amb ID `21386114815922_3` són:

```
Message [body=Cos del missatge del correu amb id
1386114815221_1386114815221_21386114815922_3, cc=Destinataris en cï¿½pia del
missatge amb id 1386114815221_1386114815221_21386114815922_3, color=0,
date=Wed Dec 04 00:53:35 CET 2013, from=Remitent del missatge amb id
1386114815221_1386114815221_21386114815922_3,
id=1386114815221_1386114815221_21386114815922_3, snippet=Primers caracters del
missatge amb id 1386114815221_1386114815221_21386114815922_3, status=0,
subject=Tema del missatge amb id 1386114815221_1386114815221_21386114815922_3,
to=Destinataris del missatge amb id
1386114815221_1386114815221_21386114815922_3]
```

La pantalla corresponent és:

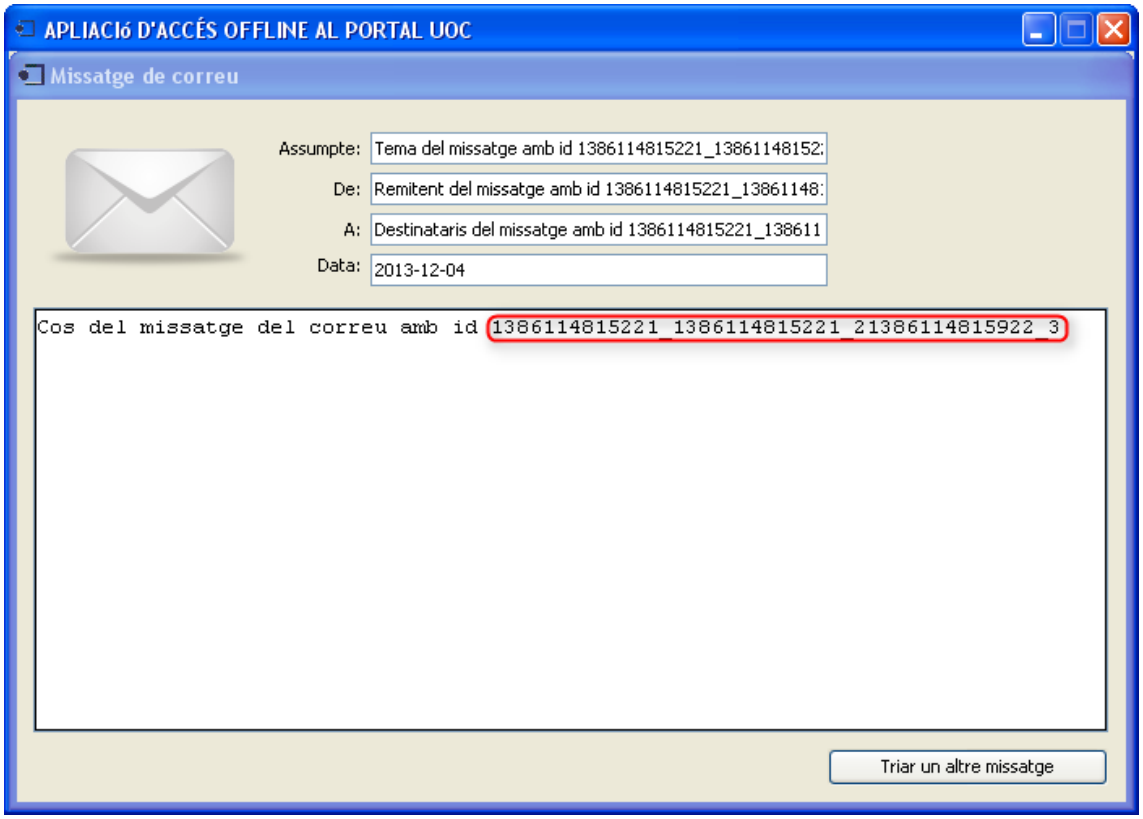

Figura 55 – Joc de proves. Consulta de correu 3.

Les dades llegides per consola corresponents al calendari són:

```
EventList [events=[
Event [end=Wed Dec 04 01:53:34 CET 2013, id=1, start=Wed Dec 04
00:53:34 CET 2013, summary=Sumary de l'event amb id 1, url=Url de
l'event amb id 1],
Event [end=Wed Dec 04 12:23:34 CET 2013, id=2, start=Wed Dec 04
11:53:34 CET 2013, summary=Sumary de l'event amb id 2, url=Url de
l'event amb id 2],
Event [end=Fri Dec 06 00:53:34 CET 2013, id=3, start=Thu Dec 05
23:53:34 CET 2013, summary=Sumary de l'event amb id 3, url=Url de
l'event amb id 3]]]
Event [end=Wed Dec 04 01:53:34 CET 2013, id=1, start=Wed Dec 04
00:53:34 CET 2013, summary=Sumary de l'event amb id 1, url=Url de
l'event amb id 1]
```

La pantalla corresponent és:

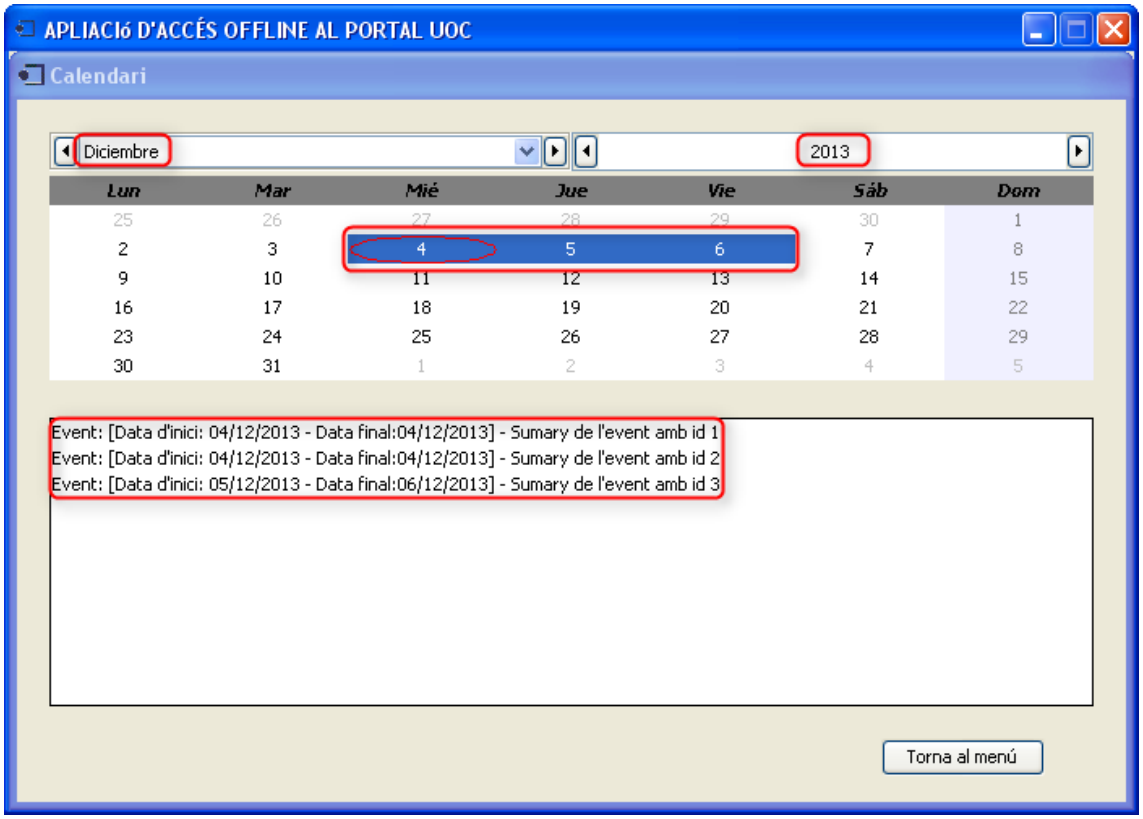

Figura 56 – Joc de proves. Consulta del calendari.

S'han llegit 4 tasques, però la quarta és la primera repetida. Per aquest motiu, només hi ha les dades de 3 tasques.

El calendari se situa automàticament al més actual. A més, surten marcades en color blau les dates on hi ha lliuraments o inicis de tasques. La data actual surt encerclada en vermell.

Les dades llegides per consola corresponents a la consulta d'aules són:

```
ClassRoomList [classrooms=[
ClassRoom [assignments=[ESTUDIANT], boards=null, color=#308961,
fatherId=111111, id=1111112, title=Introduction to Java Programming
Aula 1],
ClassRoom [assignments=[ESTUDIANT], boards=null, color=#308961,
fatherId=444444, id=4444441, title=Test Main subject Aula 1],
ClassRoom [assignments=[ESTUDIANT], boards=null, color=#308961,
fatherId=555555, id=5555551, title=Test Lab subject Aula 1]]]
```

La pantalla corresponent és:

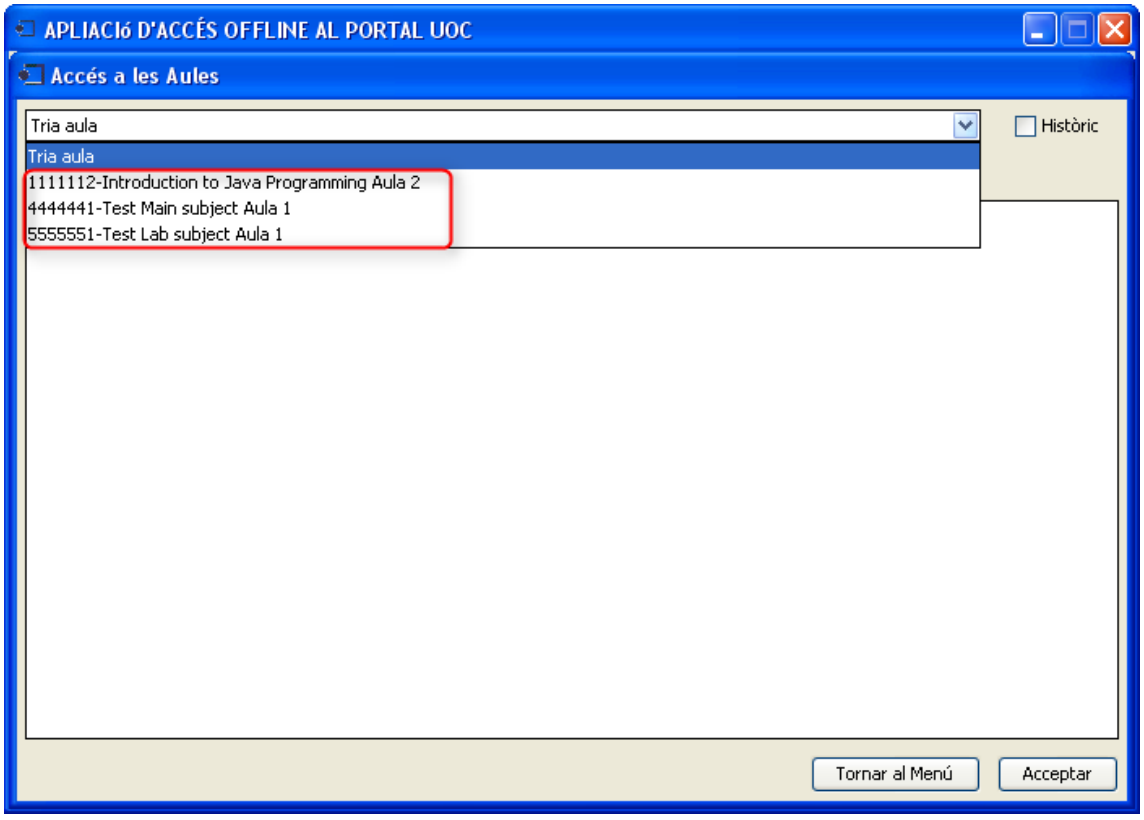

Figura 57 – Joc de proves. Consulta d'aules.

Les dades llegides per consola corresponents a la consulta dels recursos de l'aula amb ID `4444441` són:

```
BoardList [boards=[
Board [code=Codi del recurs amb id 44444411386114813127, domainId=Id
de l'aula on estï¿½ aquest recurs, id=44444411386114813127,
subtype=subtipus, title=Tï¿½tol del recurs amb id
44444411386114813127, totalMessages=104, unreadMessages=83],
Board [code=Codi del recurs amb id 44444411386114813127, domainId=Id
de l'aula on estï¿½ aquest recurs, id=44444411386114813127,
subtype=subtipus, title=Tï¿½tol del recurs amb id
44444411386114813127, totalMessages=131, unreadMessages=96]]]
```

La pantalla corresponent és:

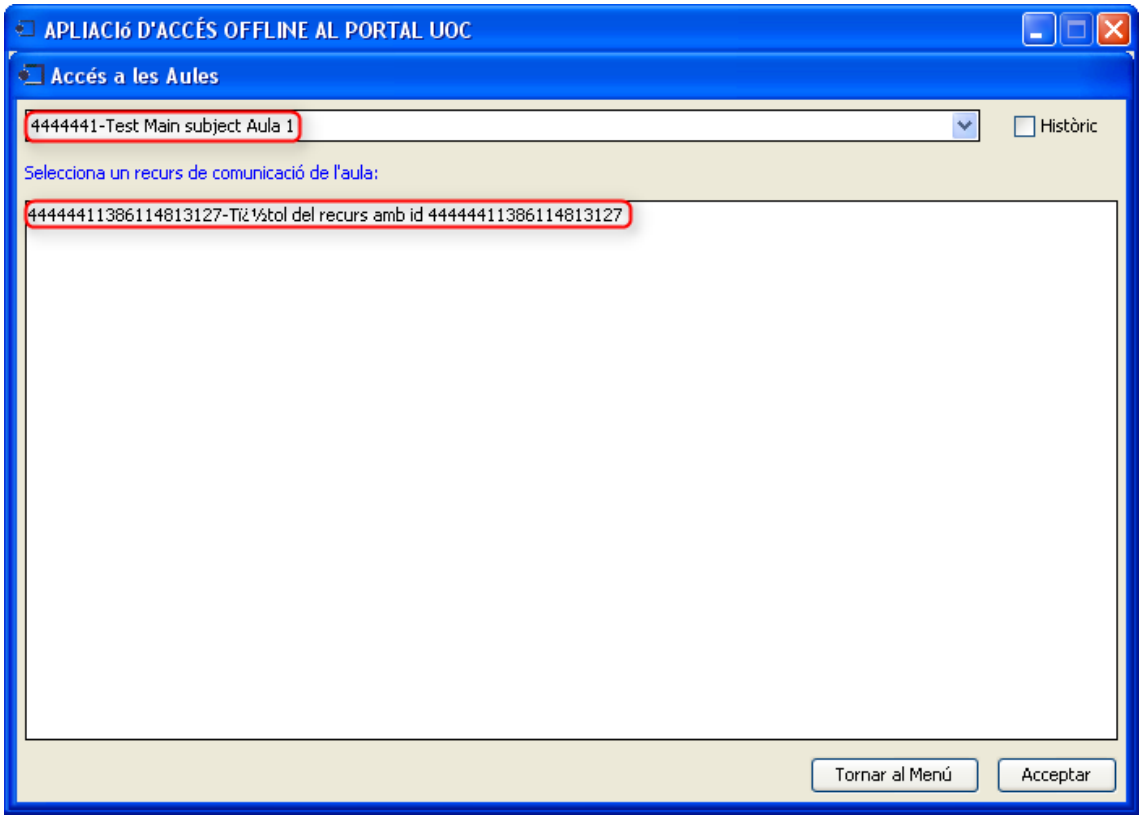

Figura 58 – Joc de proves. Consulta del recursos d'una aula

Tot i que s'han llegit dos recursos, només se n'ha registrar un. El motiu rau en el fet que els dos recursos tenen el mateix *id*.

Les dades llegides per consola corresponents als missatges del recurs amb ID `44444411386114813127` són:

```
Read: MessageList [messages=[
Message [body=Cos del missatge del correu amb id
44444411386114813127_1386114813458_11386114813458_1, cc=Destinataris en cï¿½pia del
missatge amb id 44444411386114813127_1386114813458_11386114813458_1, color=0, date=Wed
Dec 04 00:53:33 CET 2013, from=Remitent del missatge amb id
44444411386114813127_1386114813458_11386114813458_1,
id=44444411386114813127_1386114813458_11386114813458_1, snippet=Primers caracters del
missatge amb id 44444411386114813127_1386114813458_11386114813458_1, status=0,
subject=Tema del missatge amb id 44444411386114813127_1386114813458_11386114813458_1,
to=Destinataris del missatge amb id
44444411386114813127_1386114813458_11386114813458_1],
Message [body=Cos del missatge del correu amb id
44444411386114813127_1386114813458_11386114813458_2, cc=Destinataris en cï¿½pia del
missatge amb id 44444411386114813127_1386114813458_11386114813458_2, color=0, date=Wed
Dec 04 00:53:33 CET 2013, from=Remitent del missatge amb id
44444411386114813127_1386114813458_11386114813458_2,
id=44444411386114813127_1386114813458_11386114813458_2, snippet=Primers caracters del
missatge amb id 44444411386114813127_1386114813458_11386114813458_2, status=0,
subject=Tema del missatge amb id 44444411386114813127_1386114813458_11386114813458_2,
to=Destinataris del missatge amb id
44444411386114813127_1386114813458_11386114813458_2],
Message [body=Cos del missatge del correu amb id
44444411386114813127_1386114813458_11386114813458_3, cc=Destinataris en cï¿½pia del
missatge amb id 44444411386114813127_1386114813458_11386114813458_3, color=0, date=Wed
Dec 04 00:53:33 CET 2013, from=Remitent del missatge amb id
44444411386114813127_1386114813458_11386114813458_3,
id=44444411386114813127_1386114813458_11386114813458_3, snippet=Primers caracters del
missatge amb id 44444411386114813127_1386114813458_11386114813458_3, status=0,
subject=Tema del missatge amb id 44444411386114813127_1386114813458_11386114813458_3,
to=Destinataris del missatge amb id
44444411386114813127_1386114813458_11386114813458_3]]]
```

La pantalla corresponent és:

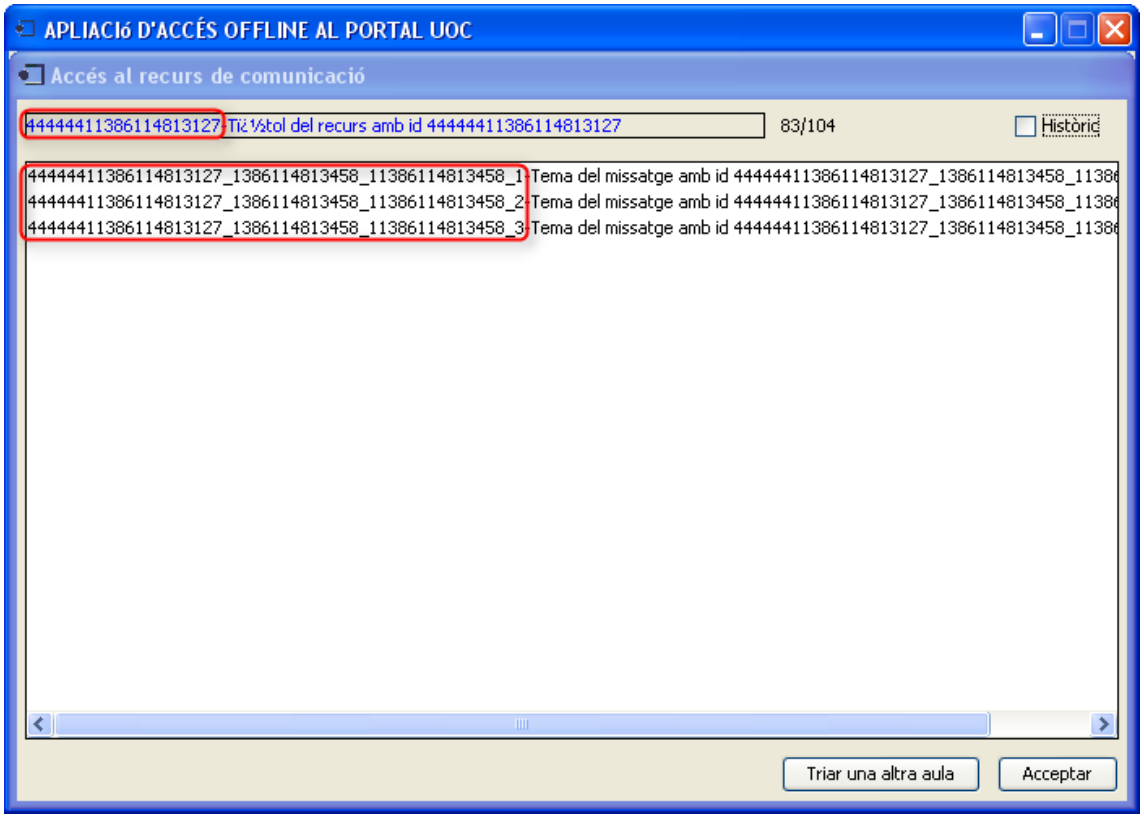

Figura 59 – Joc de proves. Consulta dels missatges d'un recurs.

Les dades llegides per consola corresponents al missatge amb l'id 44444411386114813127_1386114813458_11386114813458_2 són:

```
Message [body=Cos del missatge del correu amb id 44444411386114813127_1386114813458_11386114813458_2, cc=Destinataris en cï¿½pia del missatge amb id 44444411386114813127_1386114813458_11386114813458_2, color=0, date=Wed Dec 04 00:53:33 CET 2013, from=Remitent del missatge amb id 44444411386114813127 1386114813458 11386114813458 2, id=44444411386114813127_1386114813458_11386114813458_2, snippet=Primers caracters del missatge amb id 44444411386114813127_1386114813458_11386114813458_2, status=0, subject=Tema del missatge amb id 44444411386114813127_1386114813458_11386114813458_2, to=Destinataris del missatge amb id 44444411386114813127_1386114813458_11386114813458_2]
```

La pantalla corresponent és:

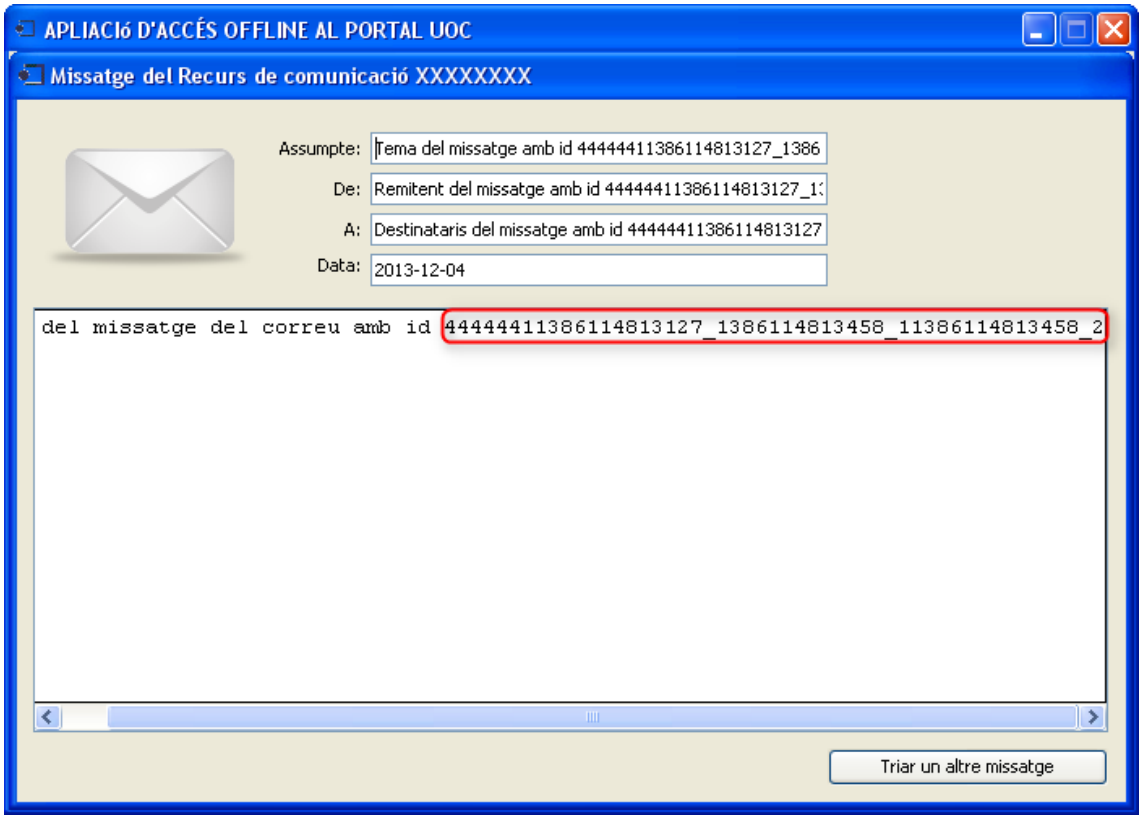

Figura 60 – Joc de proves. Dades d'un missatge.

5.11 Dificultats i problemes.

Les principals dificultats del projecte fan referència a l'ús de tecnologies que fins ara no havia utilitzat. L'autenticació *OAuth* i les consultes *JSON* en serien dos exemples clars. Tot i així, un cop analitzades aquestes tecnologies no he tingut gairebé problemes en el desenvolupament del projecte.

En canvi, sí que he tingut problemes, sobretot, amb les dades que configuren el joc de proves de l'API. Tot seguit les enumero:

1. L'autenticació a la UOC s'efectua amb les credencials vàlides d'un usuari "real" de la UOC. Un cop l'usuari s'ha autenticat, la resposta de les consultes no es corresponen amb les dades de l'usuari.

   Després de fer una consulta al tutor, em comunica que l'API no pot retornar les dades reals de l'usuari per un tema de confidencialitat de les dades personals.

   Cal recordar que el projecte enregistra localment les dades. Donat que aquestes són diferents, m'he vist obligat a no vincular l'usuari que s'autentica amb les dades de retorn de les consultes.

2. Cada cop que l'usuari executa el programa i s'autentica al campus, les dades retornades per les consultes són diferents, tot i ser el mateix usuari. No puc contrastar les dades consultades a l'API en diferents execucions del programa.

3. Sovint, els identificadors es repeteixen. Evidentment, a la base de dades local només permeto enregistrar un cop les dades amb identificadors repetits. En alguns casos, no es corresponen les dades llegides amb les dades localment enregistrades.

   Nota: Podem trobar-ne exemples en el capítol "Joc de proves" d'aquest mateix document. Val a dir, que els programadors que mantenen l'API van intentar millorat aquest aspecte, però encara hi ha alguns casos on els identificadors es repeteixen.

4. En els recursos de comunicació hi ha dos dades numèriques que emmagatzemen el nombre de missatges llegits i el nombre de missatges totals. Aquests números no es corresponen amb el nombre real de missatges. M'he limitat a mostrar per pantalla aquestes dades tot i no ser reals.

5. No he disposat ni dels tipus ni de la mida dels camps de les taules originals de la UOC. Per aquest motiu, no puc assegurar que la seva mida sigui vàlida al 100%. Només puc assegurar que funciona perfectament amb les dades de l'entorn actual de proves.

6. El meu projecte havia de llegir les dades novament incorporades al campus per informar-ne a l'usuari. Per exemple, fer la lectura dels missatges no llegits. S'havien de llegir aquests tipus de missatges, marcar-los com a llegits al campus; i enregistrar-los localment com a no llegits. Un cop l'usuari els llegís de forma local, es marcarien com a llegits.
   Aquest procés, necessita poder canviar l'estat dels missatges del campus de no llegits a llegits. Vaig preguntar al tutor si hi havia alguna forma de fer-ho. La resposta va ser que quan es consultava un missatge, automàticament quedava marcat com a llegit.

   Després d'haver-ho provat, no funcionava. Després de consultar-ho novament, la resposta del tutor va ser que era un *bug* de l'API.

   Per tant, no he pogut implementar l'opció de mostrar l'històric de missatges i, a més, en comptes de llegir els missatges no llegits, els llegeixo tots.

5.12 Canvis respecte el projecte inicial.

Tot i que els canvis s'han comentat en l'apartat anterior, m'ha semblat important fer-ne un recull:

- No he vinculat les dades de l'usuari que s'autentica amb les dades de l'usuari que retorna el campus.
- No es poden contrastar les dades consultades a l'API en diferents execucions del programa.
- No es correspon el nombre de dades llegides amb el nombre de dades localment enregistrades.
- M'he limitat a presentar per pantalla el nombre de missatges tot i no ser reals.
- No he pogut implementar l'opció de mostrar l'històric de missatges i, a més, en comptes de llegir els missatges no llegits, els llegeixo tots. L'objecte vist-i-plau que apareix a les pantalles on es mostren llistes de missatges no té funcionalitat.
- No puc assegurar al 100% que la mida dels camps sigui correcta.

5.13 Propostes de millora.

Implementar la funcionalitat de l'històric de missatges, i la lectura dels missatges no llegits. Tot i no ser exactament una millora, seria necessari per acabar la funcionalitat inicial del projecte.

També seria interessant afegir una opció per crear un arxiu històric de les dades actuals just abans d'inicialitzar un nou curs.

Una altra millora seria poder imprimir les dades consultades.

6. Conclusions

L'objectiu principal del projecte, era consultar les dades dels usuaris de la UOC mitjançant l'ús d'una API. El primer repte era analitzar amb detall la documentació tècnica de *l'API* i trobar les tecnologies que permetessin assolir aquest objectiu.

El sistema d'autenticació *OAuth* que utilitza *l'API,* era el primer escull a franquejar. La solució venia de la mà de les llibreries *Oauth2-client*. Respecte al sistema d'autenticació he de dir que sembla força segur; però, caldria estar alerta de possibles forats de seguretat per suplantació de *token*.

Un altre objectiu important era decidir com seria l'intercanvi de dades en les consultes. L'API oferia dues solucions: JSON i SOAP. Pel tipus de dades la millor elecció era el format lleuger JSON.

Aquestes i altres decisions tecnològiques m'han permès adquirir nous coneixements i aprofundir en altres que ja coneixia.

En aquest projecte he posat en pràctica molts dels conceptes tractats al llarg del Grau, i n'he comprovat la validesa.

Certament, he tingut alguns problemes amb la coherència de les dades consultades al campus, però, també m'han obligat a prendre decisions i a trobar solucions als reptes que s'han anat presentant.

Queda pendent la implementació de l'històric de missatges i la consulta al campus dels missatges no llegits. Ara bé, donat que el projecte es desenvolupa en un entorn de proves, no ha estat possible la seva implementació.

7. Glossari

API	***A**pplication **P**rogramming **I**nterface*. És el conjunt de funcions i procediments (o mètodes en la programació orientada a objectes) que ofereix certa biblioteca per ser utilitzada en un altre programari com una capa d'abstracció.
REST	És una tècnica d'arquitectura de programari per a sistemes hipermèdia distribuïts como la *World Wide Web*. Està basat en un protocol client/servidor sense estat.
OAuth	És un protocol obert, proposat per *Blaine Cook* i *Chris Messina*, que permet l'autorització segura d'una *API* de mode estàndard i simple, per aplicacions d'escriptori, de mòbils i de web.
JSON	Acrònim de *JavaScript Object Notation*. És un format lleuger per a l'intercanvi de dades. *JSON* és un subconjunt de la notació literal d'objectes de *JavaScript* que no requereixen de l'ús *d'XML*
XML	Acrònim d'*e**X**tensible **M**arkup **L**anguage* ('llenguaje de marques extensible'). És un llenguatge de marques desenvolupat per la *World Wide Web Consortium* (W3C) utilitzat per a emmagatzemar dades en format llegible.
URI	Acrònim d'***Uniform Resource Identifier*** o ***URI*** (identificador uniforme de recursos). És una cadena de caràcters curta que identifica inequívocament un recurs (servei, pàgina, document, adreça de correu electrònic, enciclopèdia, etc.).
AJAX	És una tècnica de desenvolupament web per a crear aplicacions interactives. Aquestes aplicacions s'executen al costat del client, és a dir, al navegador dels usuaris mentre es manté la comunicació asíncrona amb el servidor en segon pla.
SHA-2	Conjunt de funcions *hash* de xifratge designades per la *U.S. National Security Agency* (NSA) i publicades en el 2001 per la *NIST* com a *U.S. Federal Information Processing Standard* (FIPS).

8. Bibliografia

OPEN API (2012, 4 de setembre),
"*API Main page*"
[Documentació de l'Open API de la UOC]
[Data de consulta: 24 de novembre de 2013]
<http://blogs1.uoc.es/developer/>

OAuth 2.0 (2007, 5 de setembre)
"OAuth protocol"
[*Specifications of OAuth 2.0 protocol*]
[Data de consulta: 2 d'octubre de 2013]
<http://oauth.net/2/>

Wikipedia (2013, 13 octubre**)**
"Oauth"
[Documentació sobre el protocol obert d'autorització OAuth]
[Data de consulta: 2 d'octubre de 2013]
<http://es.wikipedia.org/wiki/OAuth>

Google Developers
"Aspectes generals d'OAuth"
[Documentació sobre OAuth i Java]
[Data de consulta: 5 d'octubre de 2013]
<https://developers.google.com/appengine/docs/java/oauth/overview?hl=es>

Wikipedia (2013, 9 de març**)**
"Arquitectura REST"
[Documentació sobre l'arquitectura REST]
[Data de consulta: 6 d'octubre de 2013]
<http://ca.wikipedia.org/wiki/Representational_State_Transfer>

Oracle JAVA (2013)
"Trail: Creating a GUI With JFC/Swing"
[The Swing Tutorial]
[Data de consulta: 2 d'octubre de 2013]
<http://docs.oracle.com/javase/tutorial/uiswing/>

MySQL Open Source Database
"MySQL Documentation"
[Documentació i descàrrega de MySQL]
[Data de consulta: 2 d'octubre de 2013]
<http://dev.mysql.com/>

Moreno Vergara, N., Vallecillo Moreno, A., Romero Salguero, J.R., Durán Muñoz, **F.** (2006) *Enginyeria del programari de components i sistemes distribuïts* (1a Edició).
Barcelona: Fundació UOC. Dipòsit legal: B-28.970-2006

Campderrich Falgueras, B., Valcárcel Larriba, R. (Febrer de 2004) Enginyeria del programari orientat a l'objecte (3a Edició).
Barcelona: Fundació UOC. ISBN: 84-9788-071-4

MySQL Workbench Community.
"Database Design & Modeling "
[Documentació i descàrrega del bastiment MySQL Workbench]
[Data de consulta: 25 de novembre de 2013]
<http://dev.mysql.com/downloads/tools/workbench/>

Cryptographic Hash & SHA-3
"Standard development"
[Documentació sobre el successor d'SHA-2]
[Data de consulta: 2 de desembre de 2013]
<http://csrc.nist.gov/groups/ST/hash/sha-3/index.html>

9. Annexos

9.1 Instal·lació del programa.

Per desenvolupar i executar el programa s'han utilitzat els següents recursos:

- Maquinari:
  - Pentium 4 a 2,8 Ghz
  - 4Gb de RAM.
  - 500Gb de disc dur.
- Programari:
  - Windows XP + SP3.
  - JRE 7.
  - JDK 1.7.0
  - xulrunner-1.9.0.17.en-US.win32.zip.
  - XAMPP v 2.5.
  - Eclipse versió Galileo.
  - mysql-workbench-community-6.0.7-win32.msi.

A més, de les llibreries Java necessàries pel desenvolupament del codi. Hi ha un annex en aquest mateix document amb la relació de llibreries utilitzades.

Passos per executar el programa:

1. Extreure el contingut del següent fitxer compactat:

   xulrunner-1.9.0.17.en-US.win32.zip

2. Entrar a la carpeta "*xulrunner*" resultat del pas (1) i executar des de la terminal o consola:

   xulrunner --register-global

3. Executar al **MySQL** el següent *script* (l'script està disponible també a l'annex d'aquest document):

   scriptBD.sql

4. Executar el programa fent doble "*clic*":

   uocOffline.jar

   O bé des de la terminal amb la següent comanda:

   java -jar uocOffline.jar

Si volguéssim executar el programa des de Linux, ens caldria descarregar la versió de ***xulrunner*** corresponent des del següent enllaç:

https://developer.mozilla.org/en-US/docs/XULRunner_1.9.2_Release_Notes

A continuació, seguir les instruccions per instal·lar l'aplicació XUL al sistema.

A més, ens caldria tornar a compilar el projecte canviant la llibreria SWT actual del projecte per la seva versió de Linux corresponent. Podem descarregar-la des del següent enllaç:

http://download.eclipse.org/eclipse/downloads/drops4/R-4.2.2-201302041200/#SWT

Nota: Cal recordar que hem de tenir instal·lada la versió *JRE* 7 i triar a *l'Eclipse* el nivell de compilació 1.5 o superior.

9.2 Llibreries utilitzades.

- *WebBrowser*:
  DJNativeSwing.jar
  DJNativeSwing-SWT.jar
  swt-3.7M5-win32-win32-x86.jar
- *Oauth2-client:*
  amber-oauth2-client-0.22-incubating.jar
  amber-oauth2-common-0.22-incubating.jar
  jettison-1.2.jar
  slf4j-api-1.6.1.jar

- *httpClient:*
  commons-logging-1.1.3.jar
  fluent-hc-4.3.1.jar
  httpclient-4.3.1.jar
  httpclient-cache-4.3.1.jar
  httpcore-4.3.jar
  httpmime-4.3.1.jar

- *google gson*:
  gson-2.2.4.jar

- *Connetor JDBC per a MySQL:*
  mysql-connector-java-5.0.8-bin.jar

- *Calendar:*
  nachocalendar-0.23.jar

9.3 Mysql Workbench Community.

MySQL Workbench facilita als administradors i desenvolupadors un entorn integrat d'eines que permeten dissenyar i modelar bases de dades. El podem descarregar des del següent enllaç:

http://dev.mysql.com/downloads/tools/workbench/

És un programari *open source* sota llicència GPL.

En aquest projecte s'ha utilitzat per dissenyar la base de dades mitjançant el model EER. Tot seguit es mostra una captura del disseny físic:

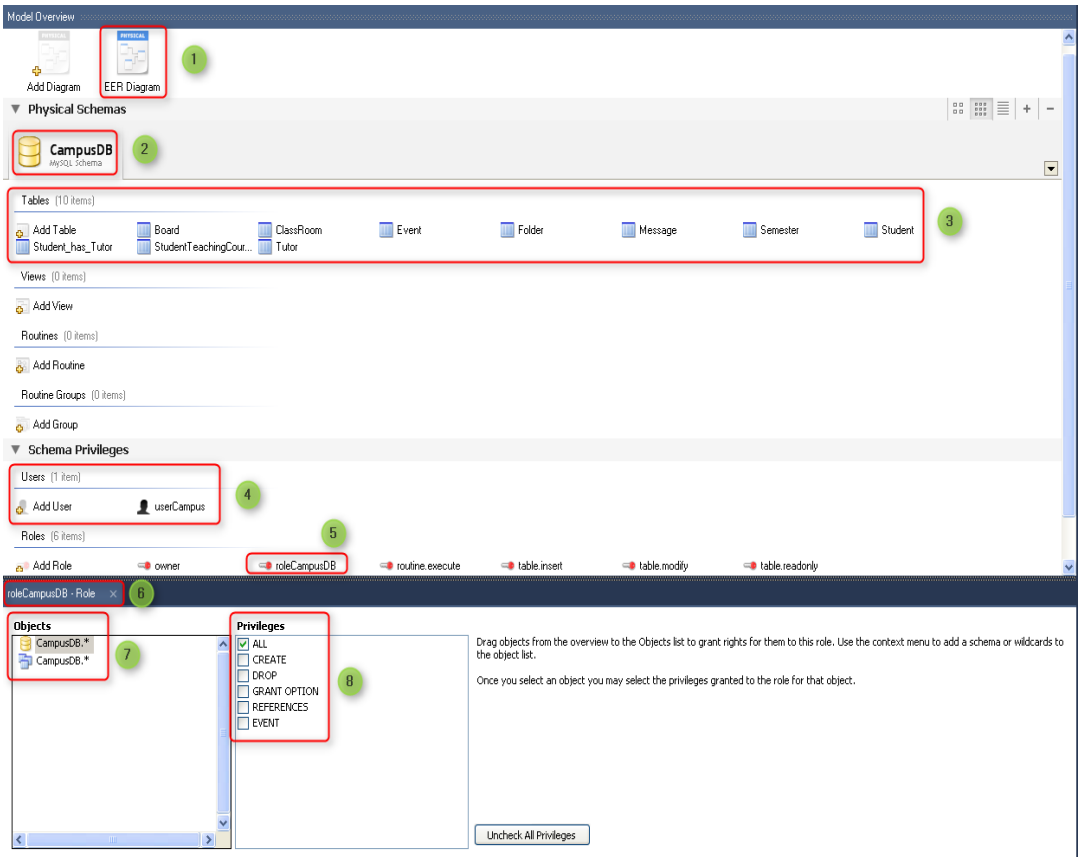

Figura 61 – Crear l'esquema físic de la base de dades CampusDB.

- En la part superior (1) permet afegir un nou diagrama EER[13].
- En el punt (2) ens permet afegir una nova base de dades, en el nostre cas "*CampusDB*".
- En el punt (3) ens permet afegir taules a la BD.
- En el punt (4) permet afegir nous usuaris, en el nostre cas s'ha afegir l'usuari *userCampus* el qual tindrà tos els privilegis sobre la BD.

[13] **E**nhanced **E**ntity-**R**elationship model.

- El punt (5) permet crear nous tipus de rols. En aquest cas s'ha afegit un nou tipus de rol "*roleCampusDB*". Aquest, dóna privilegis totals sobre la base de dades *campusDB*.
- Els punts (6,7 i 8) mostren com es configura.

Un cop creat l'esquema físic passem a crear el model EER:

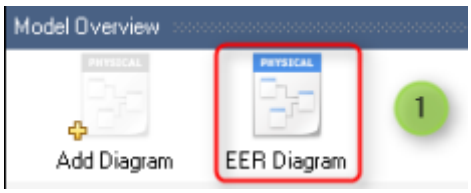

Figura 62 – Afegir diagrama EER.

La pantalla per crear el model EER és la següent:

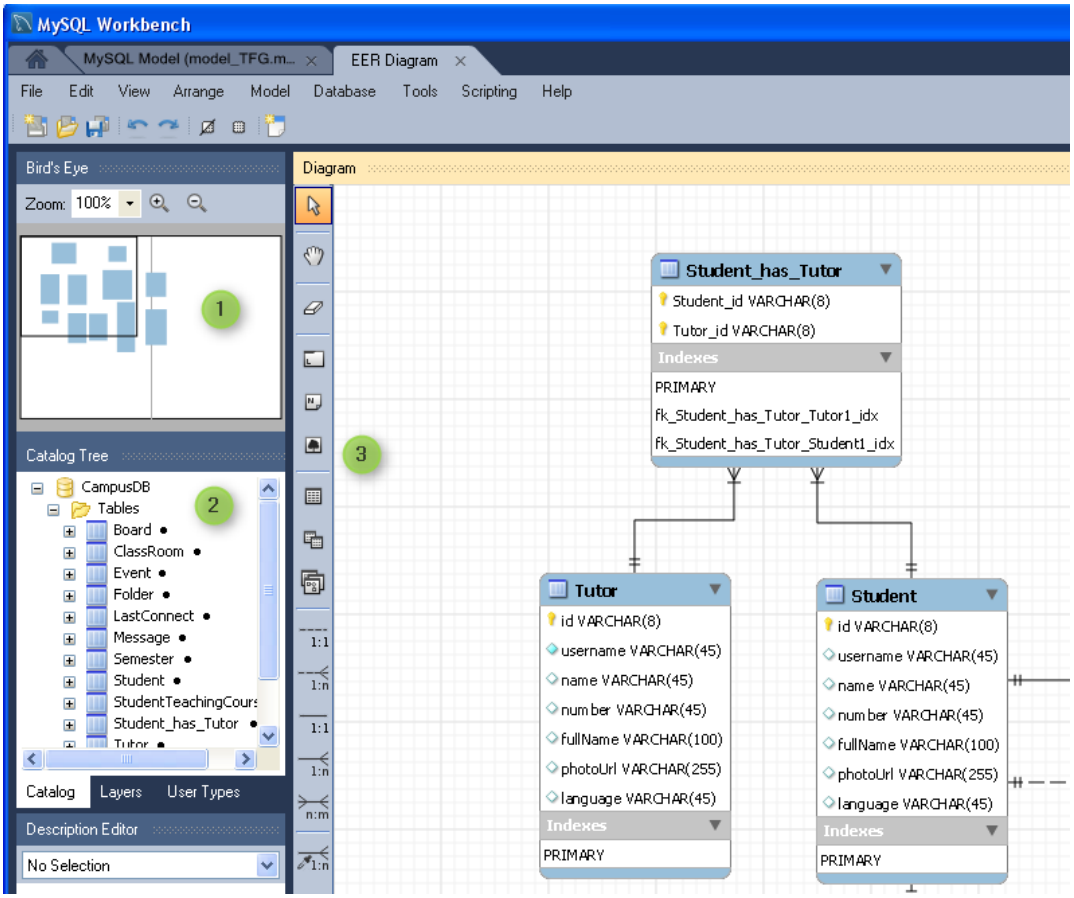

Figura 63 – Crear i editar el model EER.

Les opcions bàsiques d'edició són: La finestra zoom (1), la finestra del catàleg de la BD en format arbre (2), i el dipòsit d'eines (3).

Un cop fet el disseny podem establir relacions d'integritat referencial:

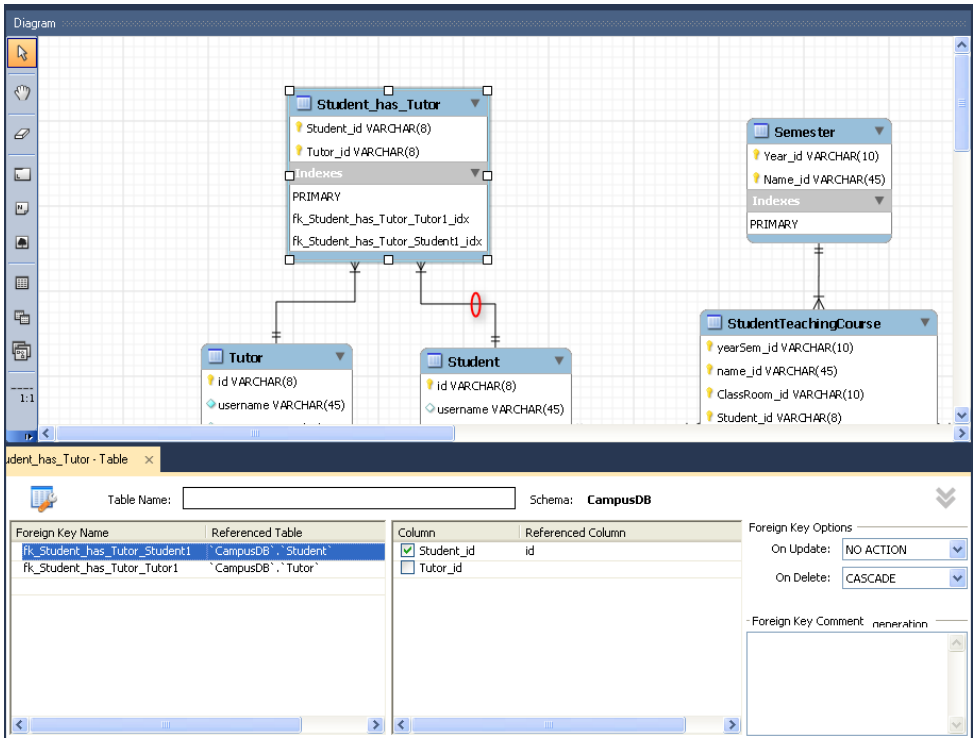

Figura 64 – Relacions d'integritat referencial.

En aquest cas podem veure com s'afegeix a la taula *Student_has_Tutor* una relació d'integritat que permet l'esborrat en cascada.

Finalment un cop acabat el disseny podem exportar el model com una consulta SQL (*File -> Export -> Sql create*):

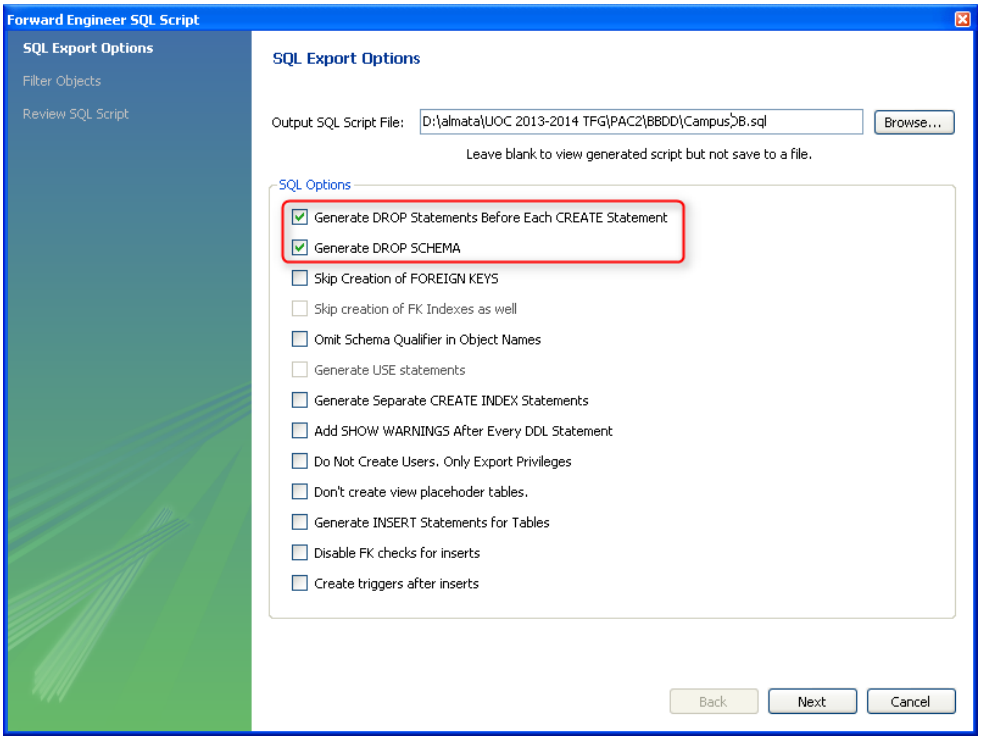

Figura 65 – Exportar model a SQL script.

9.4 *Script* SQL per crear la Base de Dades.

Tot seguit, es presenta el contingut del fitxer annex "**scriptDB.sql**". Hem d'executar aquest *script* al gestor de base de dades *MySQL* per tal de crear la base de dades de l'aplicació. Per més comoditat, es recomana fer-ho des de l'eina **phpMyAdmin**.

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';

DROP SCHEMA IF EXISTS `CampusDB` ;
CREATE SCHEMA IF NOT EXISTS `CampusDB` DEFAULT CHARACTER SET utf8
COLLATE utf8_general_ci ;
USE `CampusDB` ;

-- -----------------------------------------------------
-- Table `CampusDB`.`Tutor`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `CampusDB`.`Tutor` ;

CREATE TABLE IF NOT EXISTS `CampusDB`.`Tutor` (
  `id` VARCHAR(8) NOT NULL,
  `username` VARCHAR(45) NOT NULL,
  `name` VARCHAR(45) NULL,
  `number` VARCHAR(45) NULL,
  `fullName` VARCHAR(100) NULL,
  `photoUrl` VARCHAR(255) NULL,
  `language` VARCHAR(45) NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `CampusDB`.`Student`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `CampusDB`.`Student` ;

CREATE TABLE IF NOT EXISTS `CampusDB`.`Student` (
  `id` VARCHAR(8) NOT NULL,
  `username` VARCHAR(45) NULL,
  `name` VARCHAR(45) NULL,
  `number` VARCHAR(45) NULL,
  `fullName` VARCHAR(100) NULL,
  `photoUrl` VARCHAR(255) NULL,
  `language` VARCHAR(45) NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `CampusDB`.`Event`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `CampusDB`.`Event` ;

CREATE TABLE IF NOT EXISTS `CampusDB`.`Event` (
  `id` VARCHAR(10) NOT NULL,
  `url` VARCHAR(45) NULL,
```

```
  `summary` VARCHAR(45) NULL,
  `start` DATETIME NULL,
  `end` DATETIME NULL,
  `Student_id` VARCHAR(8) NOT NULL,
  PRIMARY KEY (`id`, `Student_id`),
  INDEX `fk_Event_Student1_idx` (`Student_id` ASC),
  CONSTRAINT `fk_Event_Student1`
    FOREIGN KEY (`Student_id`)
    REFERENCES `CampusDB`.`Student` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `CampusDB`.`Folder`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `CampusDB`.`Folder` ;

CREATE TABLE IF NOT EXISTS `CampusDB`.`Folder` (
  `id` VARCHAR(100) NOT NULL,
  `name` VARCHAR(100) NULL,
  `totalMessages` MEDIUMTEXT NULL,
  `unreadMessages` MEDIUMTEXT NULL,
  `Student_id` VARCHAR(8) NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_Folder_Student1_idx` (`Student_id` ASC),
  CONSTRAINT `fk_Folder_Student1`
    FOREIGN KEY (`Student_id`)
    REFERENCES `CampusDB`.`Student` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `CampusDB`.`Semester`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `CampusDB`.`Semester` ;

CREATE TABLE IF NOT EXISTS `CampusDB`.`Semester` (
  `Year_id` VARCHAR(10) NOT NULL,
  `Name_id` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`Year_id`, `Name_id`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `CampusDB`.`ClassRoom`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `CampusDB`.`ClassRoom` ;

CREATE TABLE IF NOT EXISTS `CampusDB`.`ClassRoom` (
  `id` VARCHAR(10) NOT NULL,
  `title` VARCHAR(45) NULL,
  `color` VARCHAR(45) NULL,
  `fatherId` VARCHAR(45) NULL,
  `assignments` VARCHAR(45) NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `CampusDB`.`StudentTeachingCourse`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `CampusDB`.`StudentTeachingCourse` ;

CREATE TABLE IF NOT EXISTS `CampusDB`.`StudentTeachingCourse` (
  `yearSem_id` VARCHAR(10) NOT NULL,
  `name_id` VARCHAR(45) NOT NULL,
  `ClassRoom_id` VARCHAR(10) NOT NULL,
  `Student_id` VARCHAR(8) NOT NULL,
  PRIMARY KEY (`yearSem_id`, `name_id`, `ClassRoom_id`, `Student_id`),
  INDEX `fk_StudentTeachingCourse_ClassRoom1_idx` (`ClassRoom_id`
ASC),
  INDEX `fk_StudentTeachingCourse_Semester1_idx` (`yearSem_id` ASC,
`name_id` ASC),
  INDEX `fk_StudentTeachingCourse_Student1_idx` (`Student_id` ASC),
  CONSTRAINT `fk_StudentTeachingCourse_ClassRoom1`
    FOREIGN KEY (`ClassRoom_id`)
    REFERENCES `CampusDB`.`ClassRoom` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_StudentTeachingCourse_Semester1`
    FOREIGN KEY (`yearSem_id` , `name_id`)
    REFERENCES `CampusDB`.`Semester` (`Year_id` , `Name_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_StudentTeachingCourse_Student1`
    FOREIGN KEY (`Student_id`)
    REFERENCES `CampusDB`.`Student` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `CampusDB`.`Board`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `CampusDB`.`Board` ;

CREATE TABLE IF NOT EXISTS `CampusDB`.`Board` (
  `ClassRoom_id` VARCHAR(10) NULL,
  `title` VARCHAR(256) NULL,
  `subtype` VARCHAR(45) NULL,
  `domainId` VARCHAR(256) NULL,
  `code` VARCHAR(256) NULL,
  `totalMessages` MEDIUMTEXT NULL,
  `unreadMessages` MEDIUMTEXT NULL,
  `id` VARCHAR(25) NOT NULL,
  INDEX `fk_Board_ClassRoom1_idx` (`ClassRoom_id` ASC),
  PRIMARY KEY (`id`),
  CONSTRAINT `fk_Board_ClassRoom1`
    FOREIGN KEY (`ClassRoom_id`)
    REFERENCES `CampusDB`.`ClassRoom` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `CampusDB`.`Message`
-- -----------------------------------------------------
```

```
DROP TABLE IF EXISTS `CampusDB`.`Message` ;

CREATE TABLE IF NOT EXISTS `CampusDB`.`Message` (
  `id` VARCHAR(100) NOT NULL,
  `subject` VARCHAR(256) NULL,
  `date` DATETIME NULL,
  `snippet` VARCHAR(512) NULL,
  `status` MEDIUMTEXT NULL,
  `color` MEDIUMTEXT NULL,
  `fromField` VARCHAR(256) NULL,
  `toField` VARCHAR(256) NULL,
  `cc` VARCHAR(512) NULL,
  `body` VARCHAR(4096) NULL,
  `Folder_id` VARCHAR(50) NULL,
  `Board_id` VARCHAR(50) NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_Message_Folder1_idx` (`Folder_id` ASC),
  INDEX `fk_Message_Board1_idx` (`Board_id` ASC),
  CONSTRAINT `fk_Message_Folder1`
    FOREIGN KEY (`Folder_id`)
    REFERENCES `CampusDB`.`Folder` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Message_Board1`
    FOREIGN KEY (`Board_id`)
    REFERENCES `CampusDB`.`Board` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `CampusDB`.`Student_has_Tutor`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `CampusDB`.`Student_has_Tutor` ;

CREATE TABLE IF NOT EXISTS `CampusDB`.`Student_has_Tutor` (
  `Student_id` VARCHAR(8) NOT NULL,
  `Tutor_id` VARCHAR(8) NOT NULL,
  PRIMARY KEY (`Student_id`, `Tutor_id`),
  INDEX `fk_Student_has_Tutor_Tutor1_idx` (`Tutor_id` ASC),
  INDEX `fk_Student_has_Tutor_Student1_idx` (`Student_id` ASC),
  CONSTRAINT `fk_Student_has_Tutor_Student1`
    FOREIGN KEY (`Student_id`)
    REFERENCES `CampusDB`.`Student` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Student_has_Tutor_Tutor1`
    FOREIGN KEY (`Tutor_id`)
    REFERENCES `CampusDB`.`Tutor` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `CampusDB`.`LastConnect`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `CampusDB`.`LastConnect` ;

CREATE TABLE IF NOT EXISTS `CampusDB`.`LastConnect` (
  `User_id` VARCHAR(8) NOT NULL,
  `SemesterYear_id` VARCHAR(10) NULL,
  `SemesterName_id` VARCHAR(45) NULL,
  `data` DATETIME NULL,
  PRIMARY KEY (`User_id`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `CampusDB`.`Login`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `CampusDB`.`Login` ;

CREATE TABLE IF NOT EXISTS `CampusDB`.`Login` (
  `password` VARCHAR(512) NULL)
ENGINE = InnoDB;

SET SQL_MODE = '';
GRANT USAGE ON *.* TO 'userCampus'@'localhost';
 DROP USER 'userCampus'@'localhost';
SET SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';
CREATE USER 'userCampus'@'localhost' IDENTIFIED BY 'tfguoc2013';

GRANT ALL ON CampusDB.* TO 'userCampus'@'localhost';
GRANT ALL ON TABLE CampusDB.* TO 'userCampus'@'localhost';

/*
GRANT ALL ON CampusDB.* TO 'userCampus'@'%';
GRANT ALL ON TABLE CampusDB.* TO 'userCampus'@'%';
*/

SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```