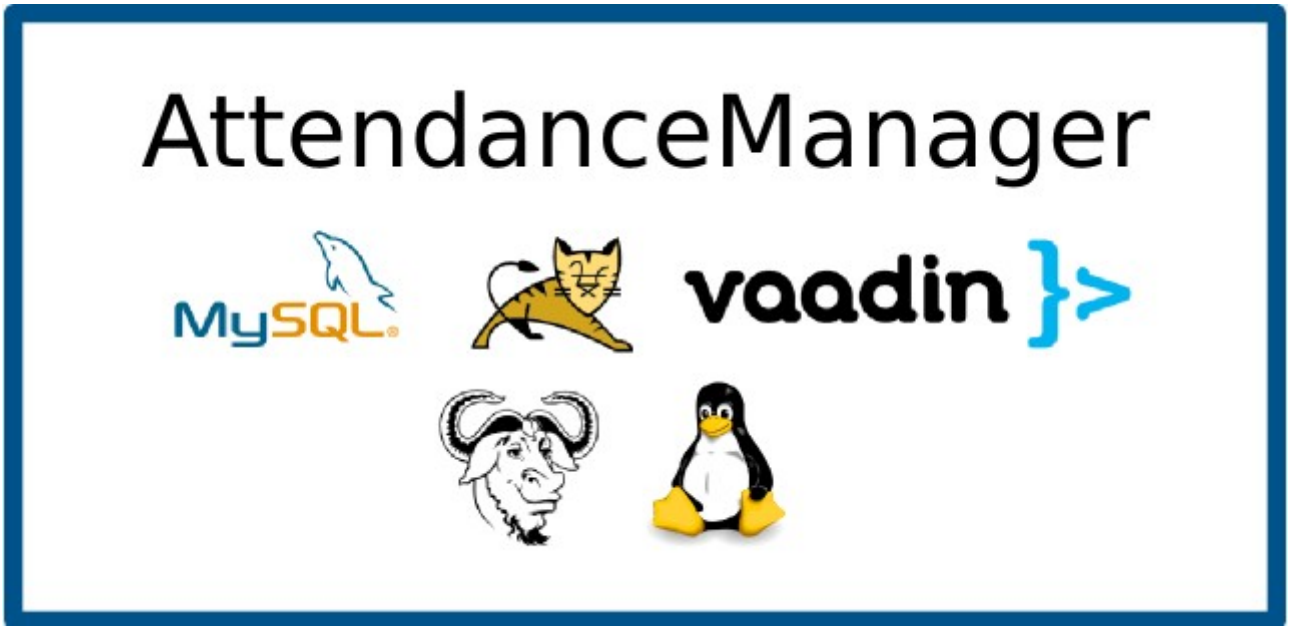
Documentació
Credit de Síntesi



Adrià Paul Cabrera Espinar

Ángel Cejudo Gómez

Raul Santos Moreno

Índex

Introducció

- **Autors**: Adrià Cabrera Espinar, Ángel Cejudo Gómez, Raul Santos Moreno.

- **Descripció**:
  - Aquest CS pretén ampliar les funcionalitats que han anat sorgint en el decurs del ús de l'aplicació AttendanceManager.
  - El llenguatge de programació utilitzat és JAVA juntament amb el API Vaadin.
  - Programat en la aplicació lliure Eclipse.

- **Millores Realitzades**:
  - Inserció de imatge per l'estudiant (modificació de la classe Student).
  - Inserció de múltiples contactes pels estudiants (modificació de la classe StudentManagerLayout).
  - Visualització de tooltip durant l'assistència (modificació de la classe TeacherMainLayout).
  - Inserció de nou tipus de falta miscel·lànea (modificació de la classe TeacherMainLayout).
  - Nova vista amb les fotos dels estudiants a l'hora de gestionar l'assistència (modificació de la classe TeacherMainLayout).
  - Resaltat en color dels estudiants que tenen faltes anteriors en una hora en concret (modificació de la classe TeacherMainLayout).
  - Posibilitat d'enviar notificacions a més d'un contacte, el qual pot o no ser actiu o per defecte (modificació de la classe CommunicationLayout).
  - Modificació de la base de dades (modificació de la classe DataBaseStorage).
  - Afegit suport d'idioma al castellà (modificació del fitxer AttendanceManagerLocalization_es.properties).
  - Creació de la classe Contact per la inserció de contactes pels estudiants.
  - Creació de la classe MyUploader per realitzar la càrrega de la foto de l'estudiant.

Gestió d'estudiants

Descripció del cas d'ús

Cas d'ús generic(ABSTRACT)

Resum de la funcionalitat:

Tractar amb els Estudiants(Altes,Baixes,Modificació,Importació).

Paper dins del treball de l'usuari:

Aquest cas d'ús és un cas d'ús principal de l'Administrador.

Actors relacionats amb el cas d'ús: L'Administrador.

Casos d'ús relacionats:

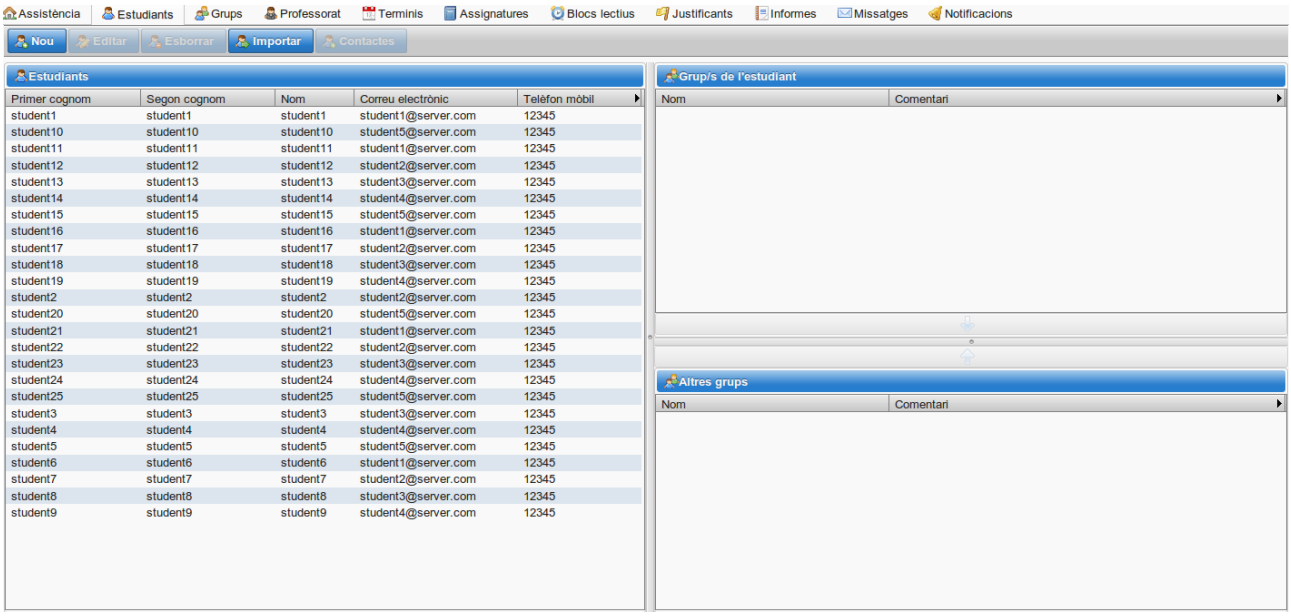
Login,Nou estudiant,Editar estudiant,Esborrar estudiant,Importar estudiant.

Precondició: Haver-se logat amb èxit.

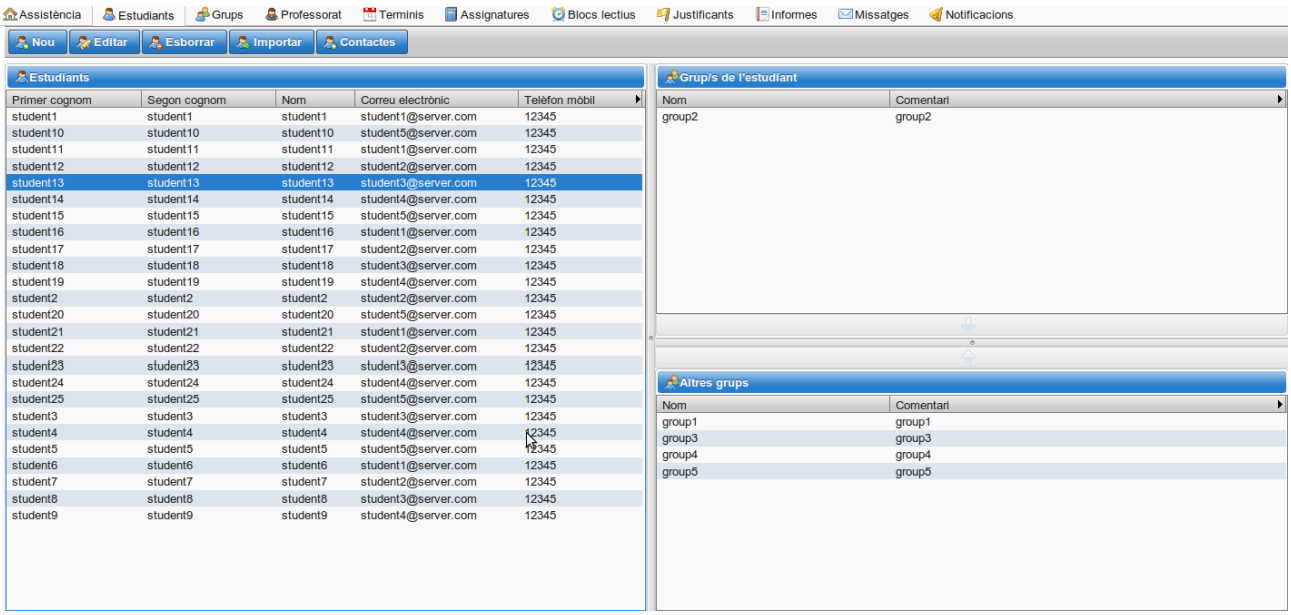
Postcondició: Actualització de la base de dades.

Funcionalitat

De inici et surten un parell de botons (Nou, Importar) seleccionables, i altres tres (Editar,Esborrar,Contactes) que no estan seleccionables,com una mena de quadre de text amb diferents columnes(Primer cognom,Segon cognom,Nom,Telefon mòbil) totes elles ordenables mitjançant un clic,al costat del nom de les columnes,trobem com una fletxa que ens deixa triar quins camps volem o no veure,dos quadres de text al costat dret,possats un damunt de l'altre un que mostra el Grup o Grups a que pertany l'estudiant(on trobem una columna per el nom i altre per els comentaris), i altre que mostra altres grups(on també trobem la columna pel nom i altre pels comentaris).



Al seleccionar qualsevol estudiant, s'activen els tres botons(Editar,Esborrar,Contactes) i passan a estar seleccionables.



Nou estudiant

Cas d'ús :

Resum de la funcionalitat:

Afegir un nou estudiant a la base de dades.

Paper dins del treball de l'usuari:

Aquest cas d'ús ès principal de l'Administrador.

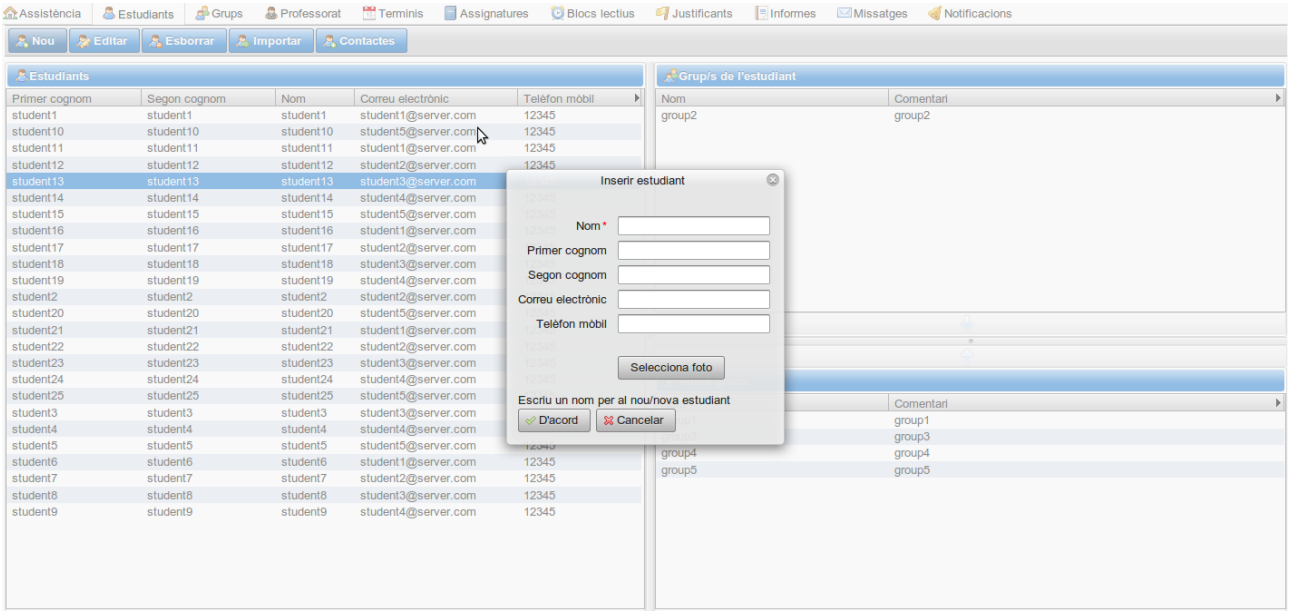
Actors relacionats amb el cas d'ús: L'Administrador.

Casos d'ús relacionats: Login.

Precondició: Haver-se logat amb èxit.

Postcondició: Actualització de la base de dades.

Quan es prem el boto (Nou),automaticament ens surt una altre finestra que ens permet introduir dades :Nom,Primer cognom,Segon cognom,Correu electrònic,Telefon mòbil, a sota un boto per carregar la foto de l'estudiant, a sota un label que diu que s'ha d'introduir un nom,i mes a sota un parell de botons mes (D'acord,Cancelar),si es prem el boto de (cancel·lar),surt de la finestra i torna a la finestra d'inici,si es prem el boto (d'acord),afegeix l'estudiant.



Quan es prem el boto d'acord i no s'ha introduït cap text hauria de mostrar alguna informació

Importar estudiant

Cas d'ús :

Resum de la funcionalitat: Importar un fitxer que conté un o més estudiants.

Paper dins del treball de l'usuari: Aquest cas d'ús és principal de l'Administrador.

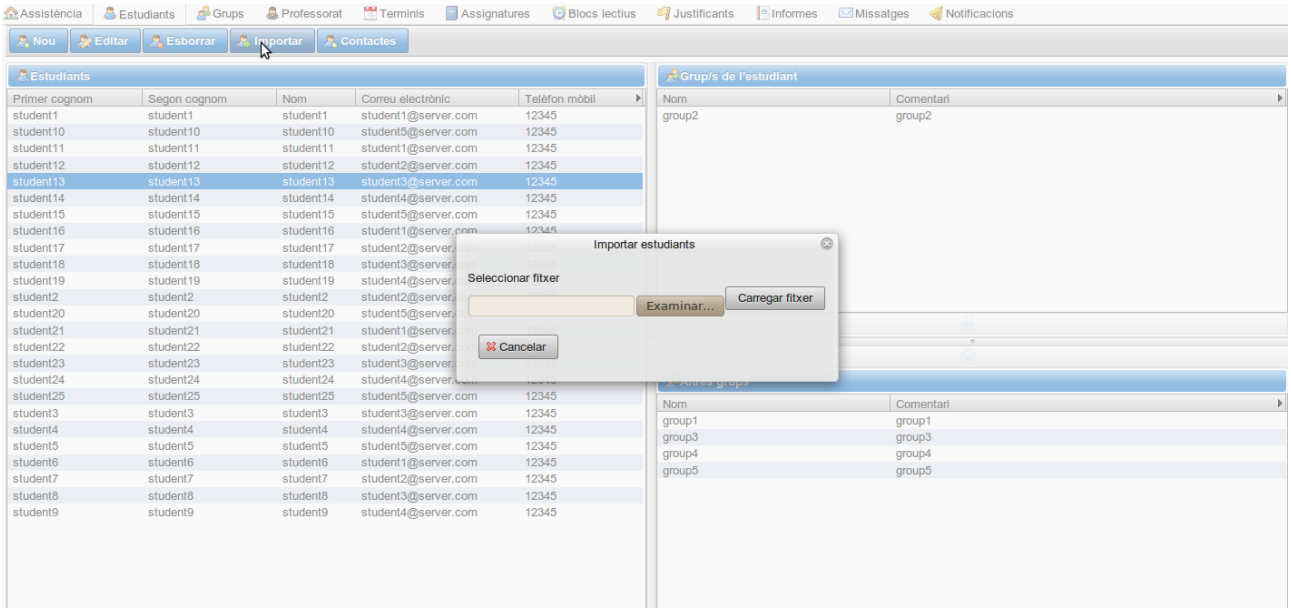
Actors relacionats amb el cas d'ús: L'Administrador.

Casos d'ús relacionats: Login.

Precondició: Haver-se logat amb èxit,el format del fitxer ha de ser (SURNAME1,SURNAME2,NAME,EMAIL,SMSPHONE) amb extensió CSV.

Postcondició: Insereix un o mes estudiants a la base de dades.

Quan es prem el boto(Importar),automaticament ens surt una altre finestra que ens permet seleccionar on es troba el fitxer que es vol importar,amb un boto per a carregar aquest fitxer.Tambe disposa d'un boto per a cancel-lar l'operació.



Quan es prem el boto de i no s'ha seleccionat cap arxiu,hauria de mostrar alguna informació

Quan seleccionem algun estudiant dels de la llista,s'obren els botons(Editar,Esborrar)i també es mostra informació referent a aquest estudiant als altres quadres(Grups del estudiants,Altres grups)

Editar estudiant

Cas d'ús :

Resum de la funcionalitat: Editar un estudiant per modificar-lo.

Paper dins del treball de l'usuari: Aquest cas d'ús és principal de l'Administrador.

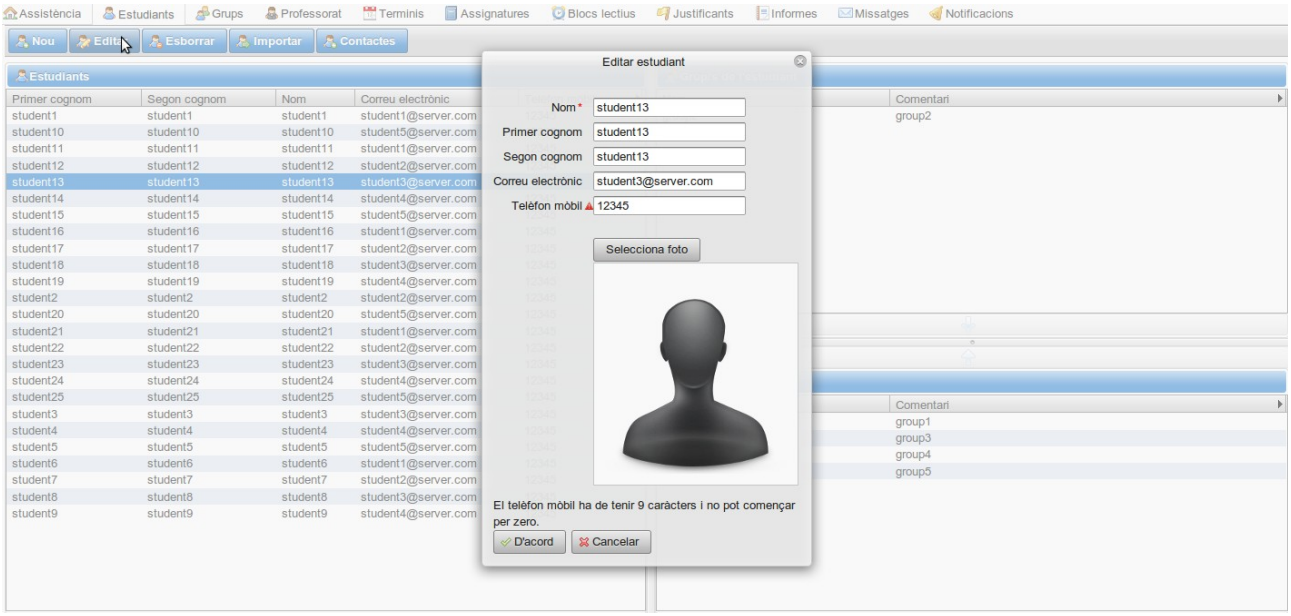
Actors relacionats amb el cas d'ús: L'Administrador.

Casos d'ús relacionats: Login,Nou estudiant.

Precondició: Haver-se logat amb èxit,haver donat d'alta algun estudiant.

Postcondició: Actualització de la base de dades.

Quan es prem el boto(Editar),sobre una finestra del mateix tipus que la de (NOU),pero en aquest cas tots els camps estan omplerts amb les dades de l'estudiant seleccionat i també la seva foto, si es canvia alguna data,quan es prem el boto (d'acord),surt un missatge a la part de sota a la dreta dient que s'ha actualitzat l'estudiant,si es prem el de (cancel-lar),surt sense fer res.



Esborrar estudiant

Cas d'ús :

Resum de la funcionalitat: Eliminar un estudiant de la base de dades.

Paper dins del treball de l'usuari: Aquest cas d'ús és principal de l'Administrador.

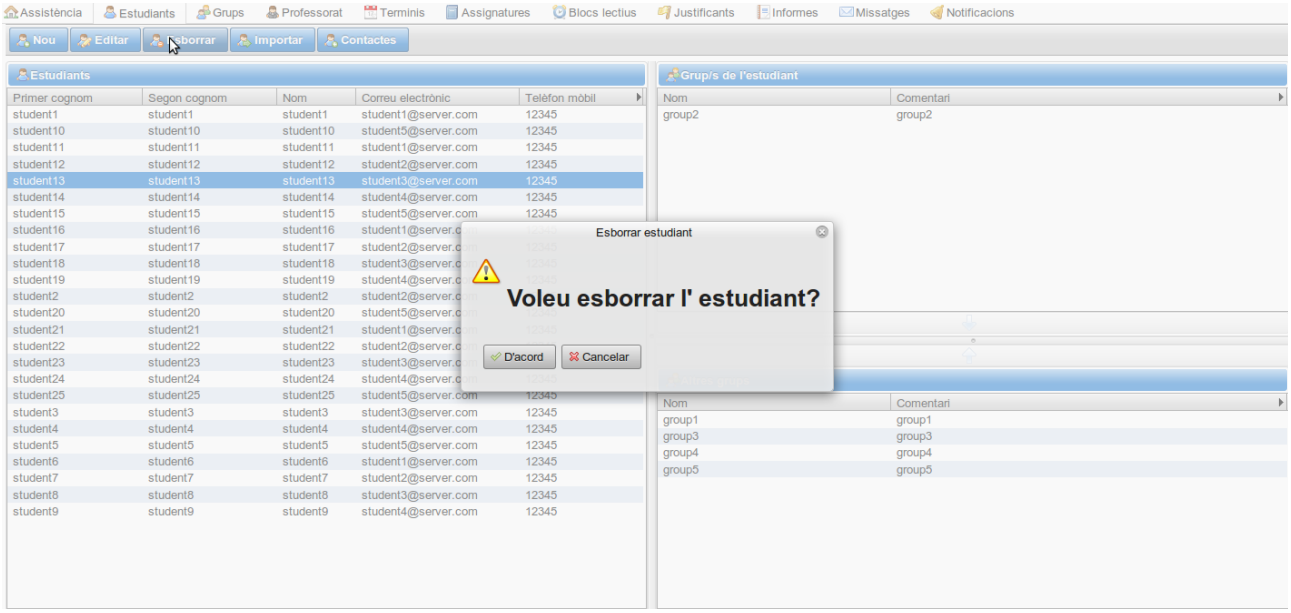
Actors relacionats amb el cas d'ús: L'Administrador.

Casos d'ús relacionats: Login,Nou estudiant.

Precondició: Haver-se logat amb èxit,haver donat d'alta l'estudiant.

Postcondició: Actualització de la base de dades.

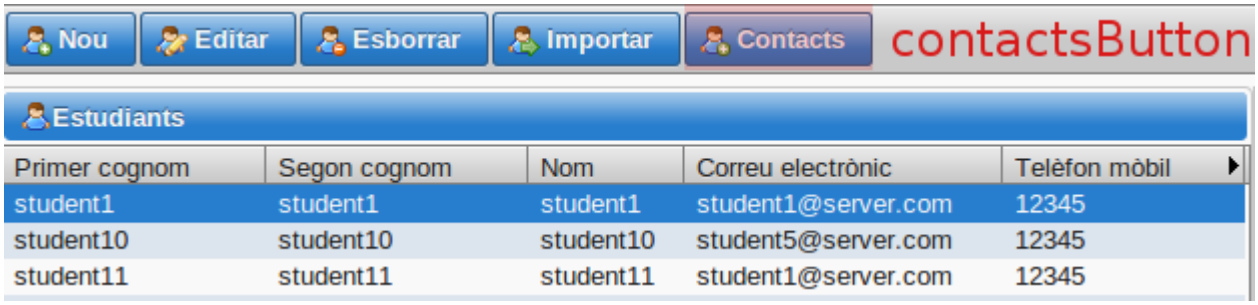
Quan es prem el boto (Esborrar),surt una finestra amb un missatge que ens pregunta si volem esborrar l'estudiant,amb dos botons (D'acord,Cancel-lar),si es prem el boto (Esborrar),s'esborra l'estudiant i ens mostra un missatge a la part de sota a la dreta i es refresca tota l'informació que hi ha als quadres de text,si es prem el boto (Cancel-lar),surt.



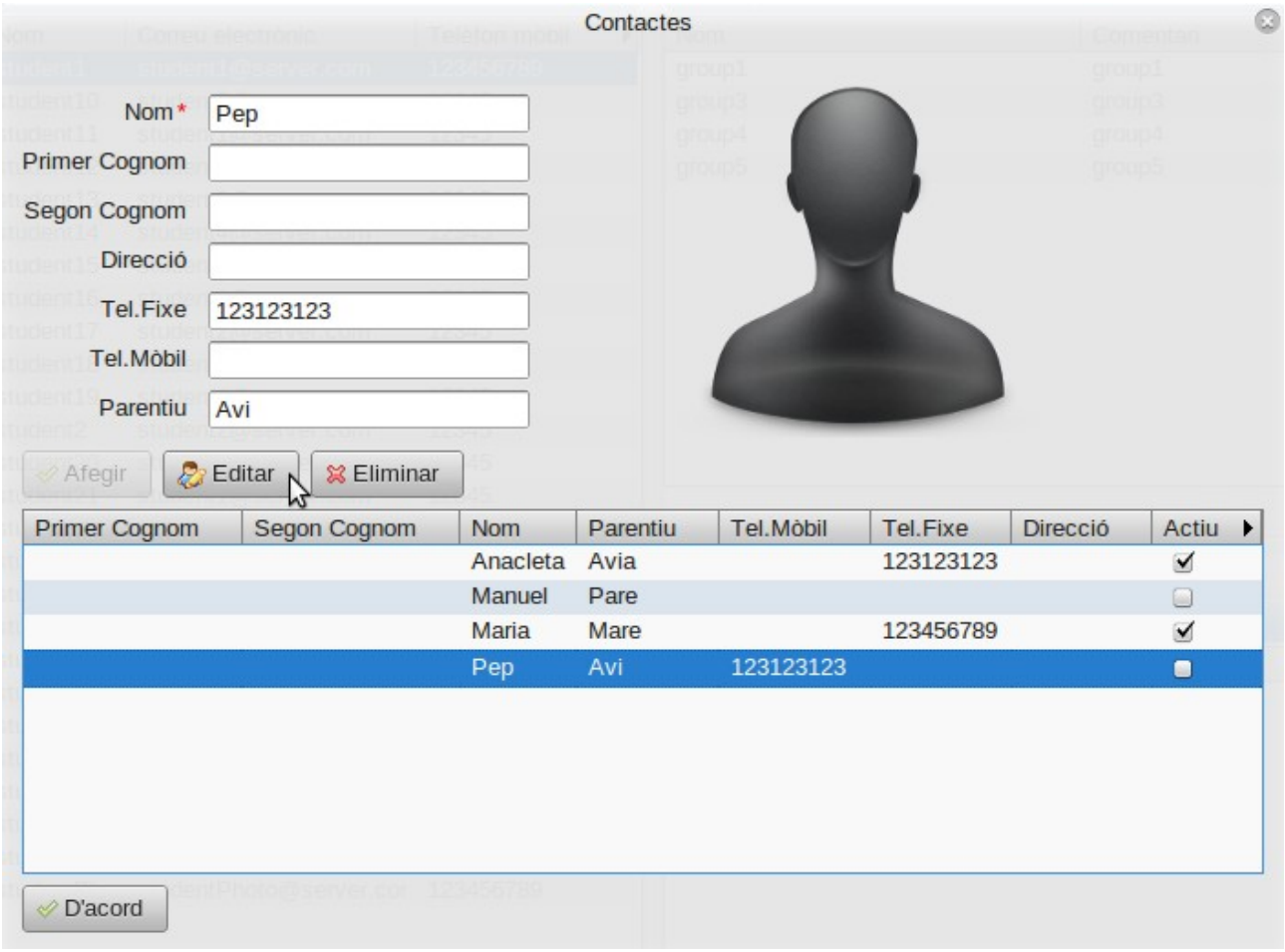
Afegir Contacte

Cas d'ús :

Resum de la funcionalitat: inserta un contacte a la base de dades. Paper dins del treball de l'usuari: Aquest cas d'ús és principal de l'Administrador. Actors relacionats amb el cas d'ús: L'Administrador. Casos d'ús relacionats: Login,Nou estudiant. Precondició: Haver-se logat amb èxit,haver donat d'alta l'estudiant. Postcondició: Actualització de la base de dades.



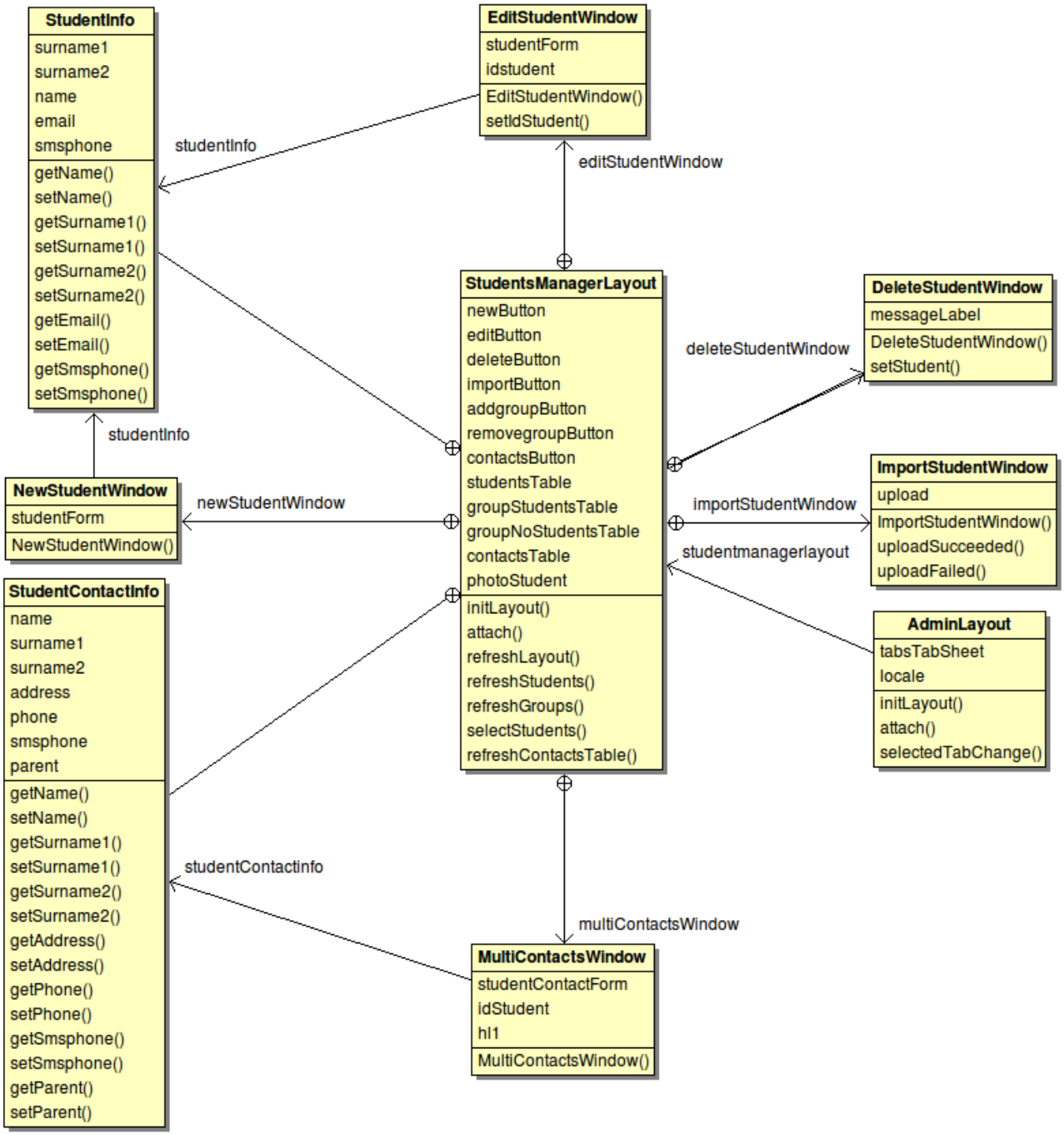
Una vegada tenim un estudiant seleccionat s'habilita el botó contactes per afegir-los al estudiant en concret.



Com podem veure, el contacte està afegit a la taula I si el seleccionem ho podem editar mitjançant el botó editar o també ho podem borrar mitjançant el botó eliminar. En la taula es mostra tota la informació del contacte, l'ultima columna anomenada Actiu indica si el contacte és prioritari a l'hora d'enviar notificacions.

Anàlisi

Diagrama de classes implicades al cas d'us



Gestió d'estudiants

Diagrama d'actors

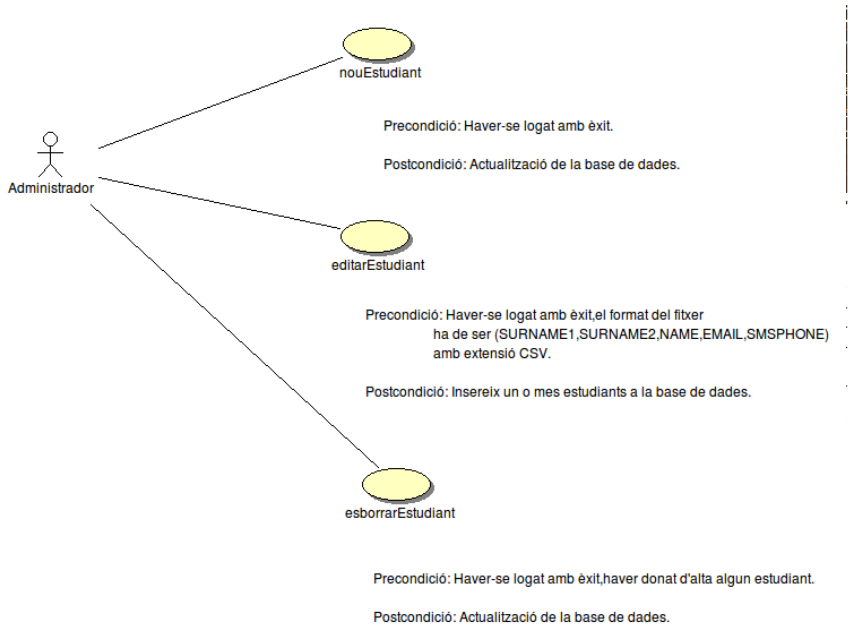
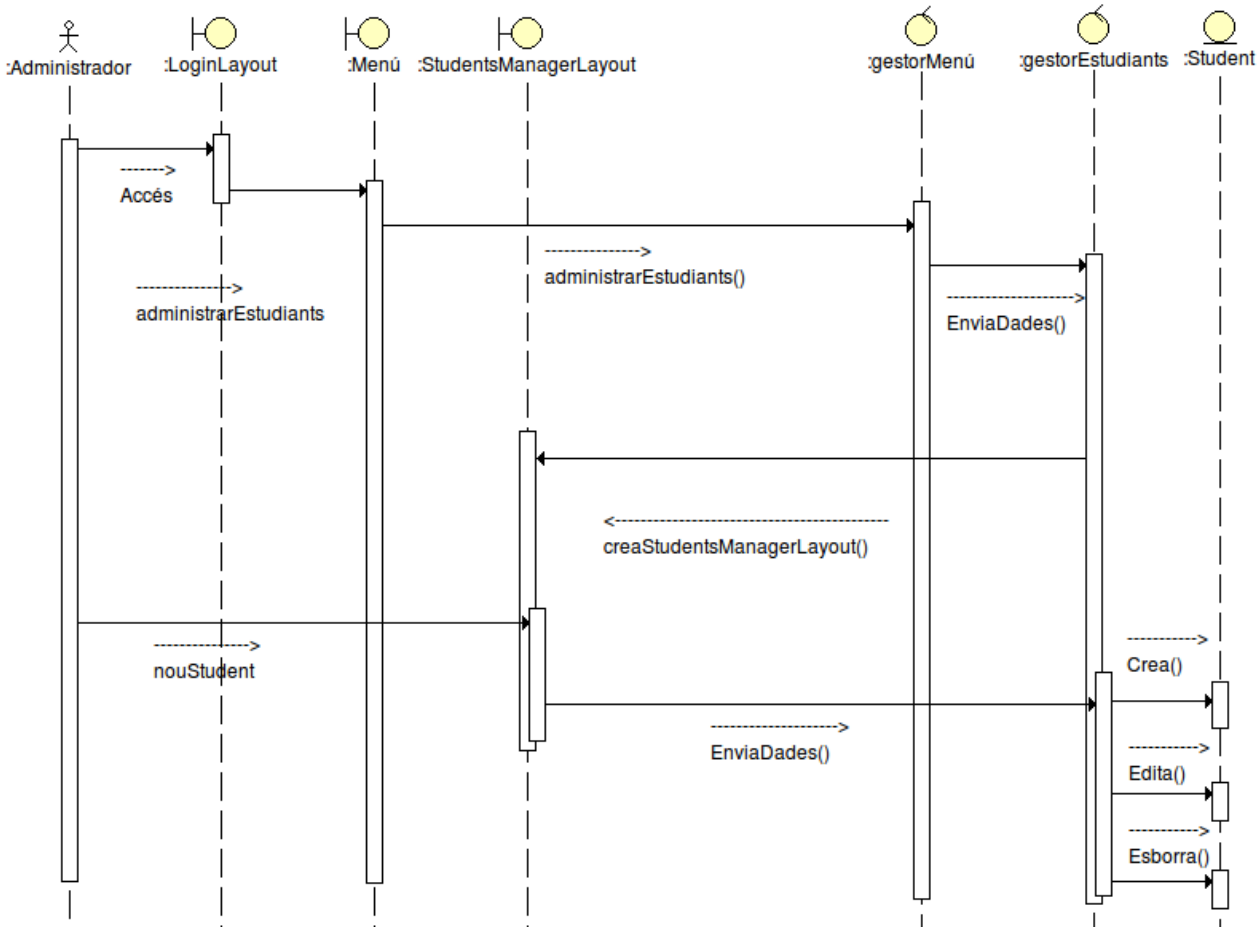


Diagrama de secuencia



Gestió de contactes

Diagrama d'actors

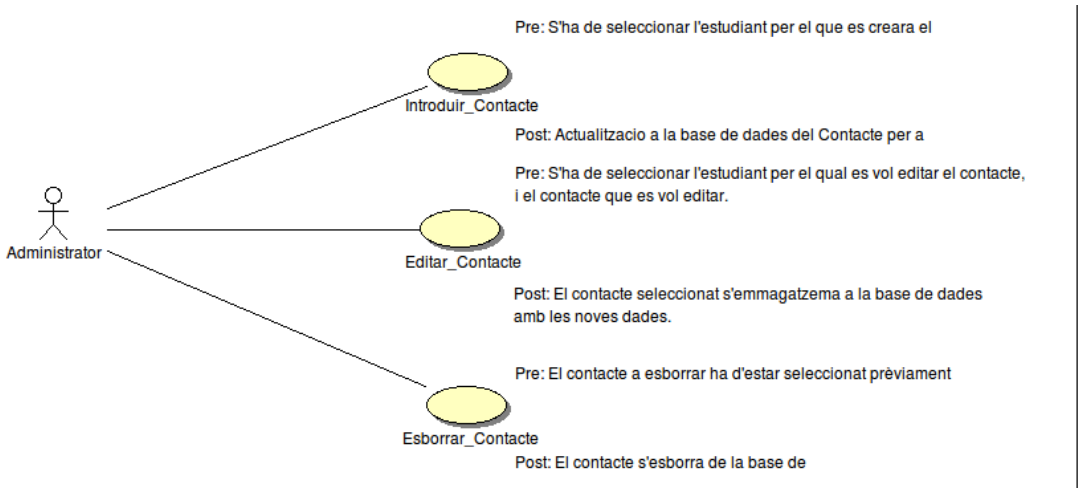
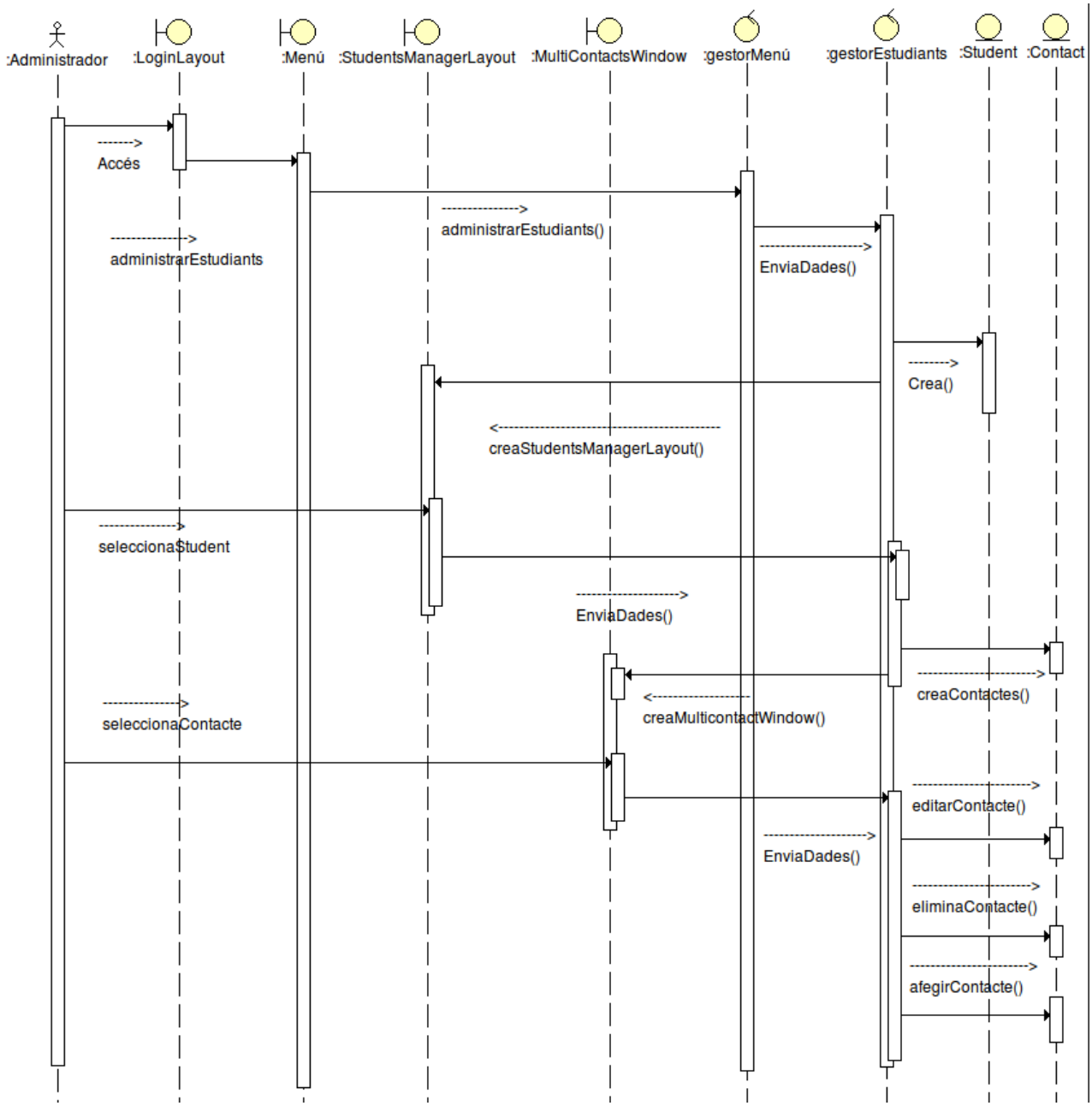


Diagrama de secuencia



Metodes de la classe

public void attach()

- Aquest mètode s'encarrega d'inicialitzar el layout.

private void initLayout()

- Construcció dels components que componen el layout (disseny).

public void refreshLayout()

- Refresca el layout mitjançant refreshStudents.

public void refreshStudents()

- Refresca els estudiants de la taula on es visualitzen.

public void refreshGroups(Integer idstudent)

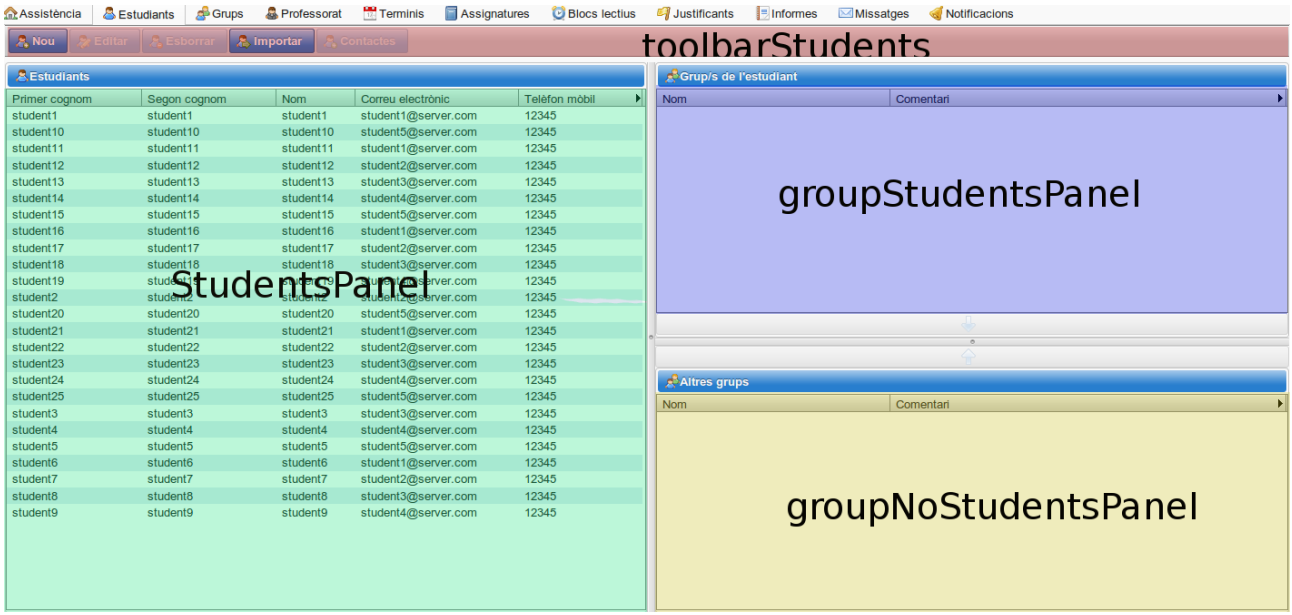
- Refresca els grups de l'estudiant seleccionat.

private void refreshContactsTable(Table contactsTable, int idStudent)

- Refresca els contactes de la taula contactsTable.

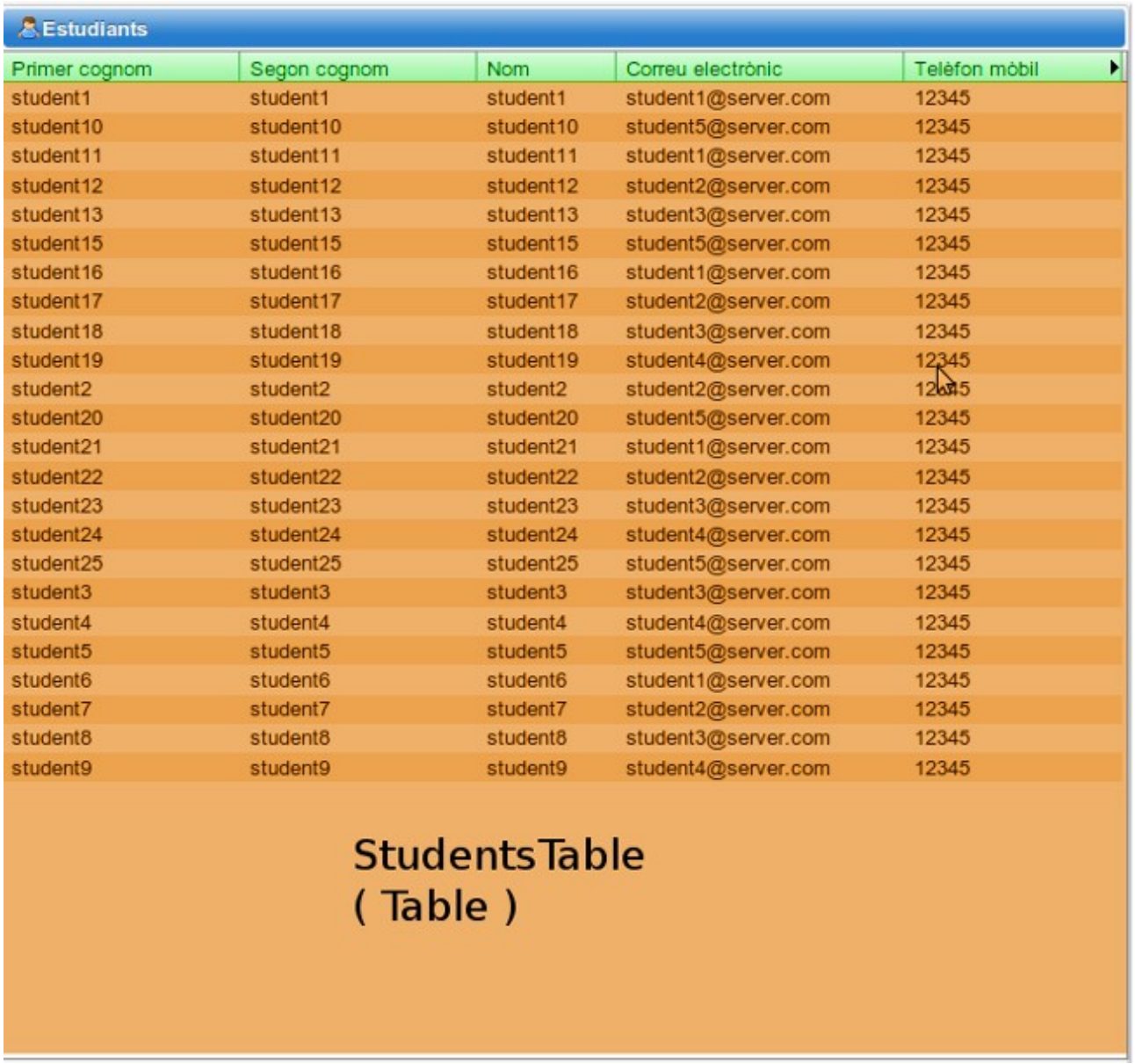
Disseny

StudentsManagerLayout (VerticalLayout)



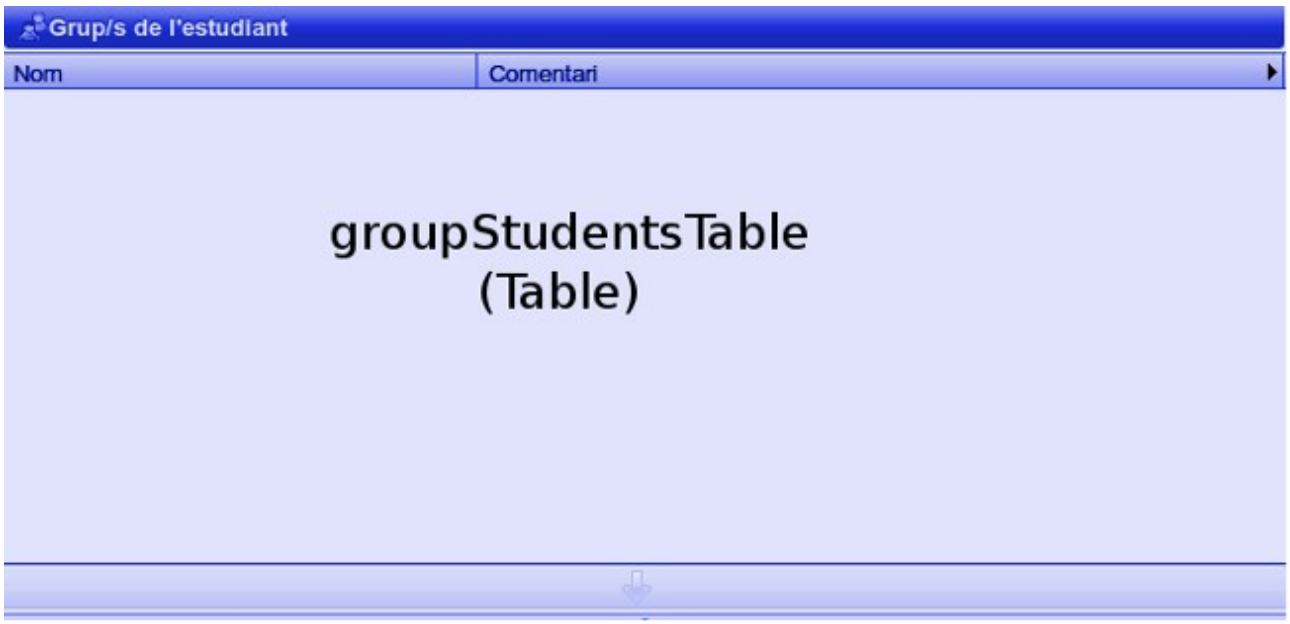
- **toolbarstudents**

- **studentsTable (Table)**



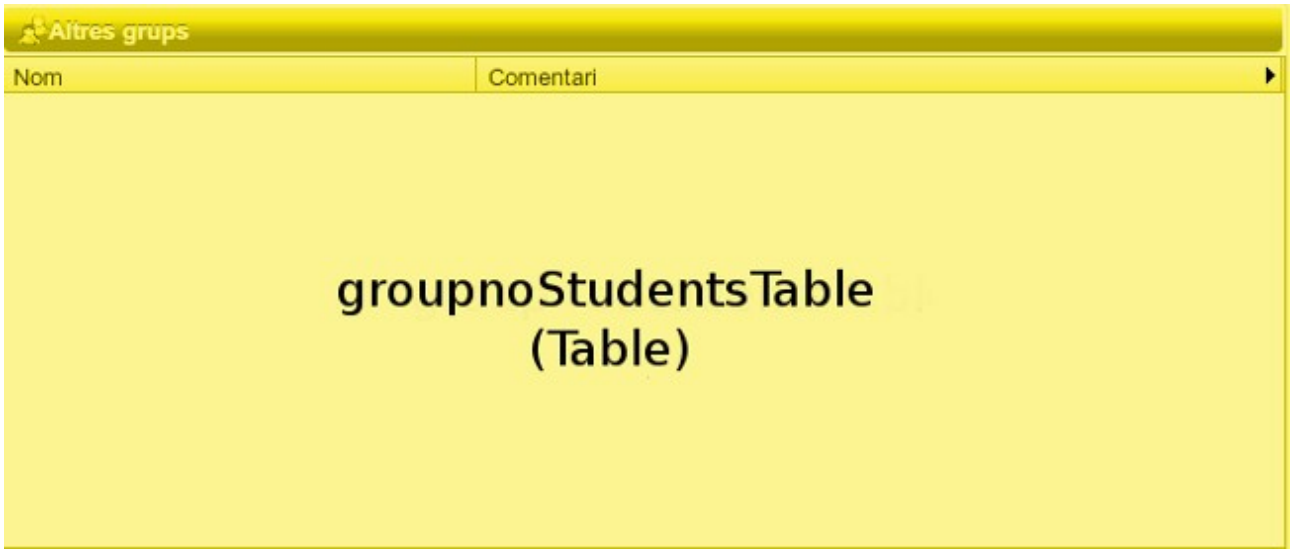
Aquesta barra conté les propietats del studentsTable

- **groupStudentsTable (Table)**



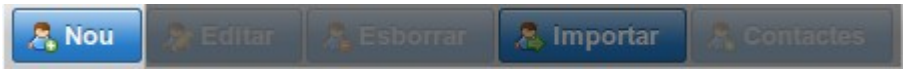
Aquesta barra conté les propietats de groupStudentTable

* **groupnoStudentsTable (Table)**



Aquesta barra conté les propietats del groupnoStudentsTable

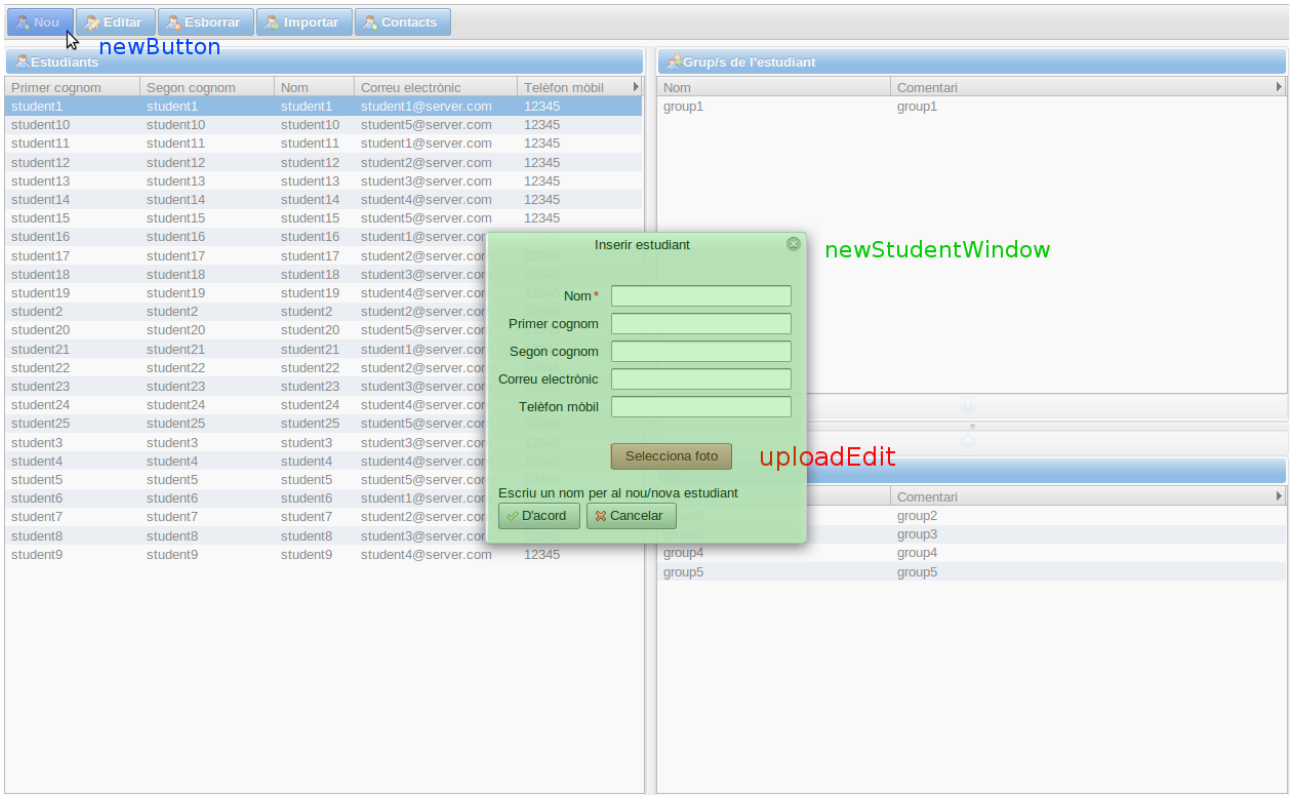
Aspectes remarcables de la implementació

Click a nou 

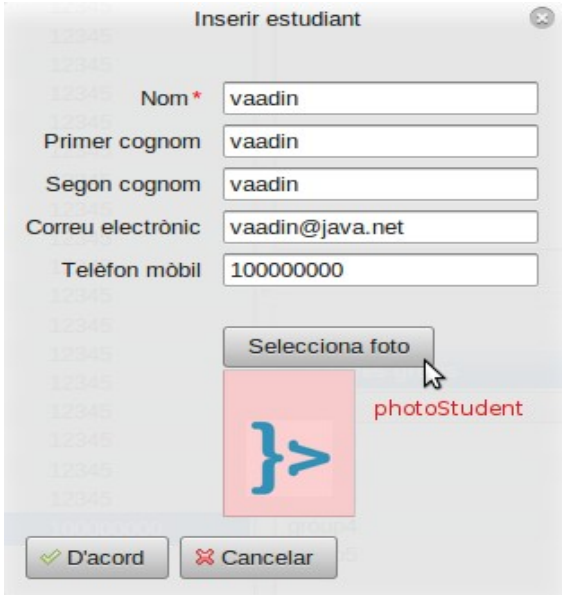
newButton.addListener(new Button.ClickListener() {

```
                @Override
                public void buttonClick(ClickEvent event) {
                        // Open window if not open already
                        if(newStudentWindow.getParent() != null) {
                                // Window already open
                        } else {
                                // open window
                                getWindow().addWindow(newStudentWindow);
                        }
                }
        });
```

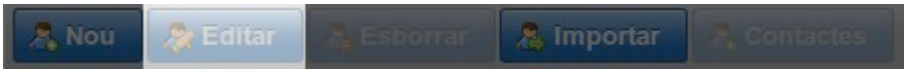
La finestra que obre el event anomenada newStudentWindow és la següent:



Una vegada afegim la foto es mostrarà de la següent manera:

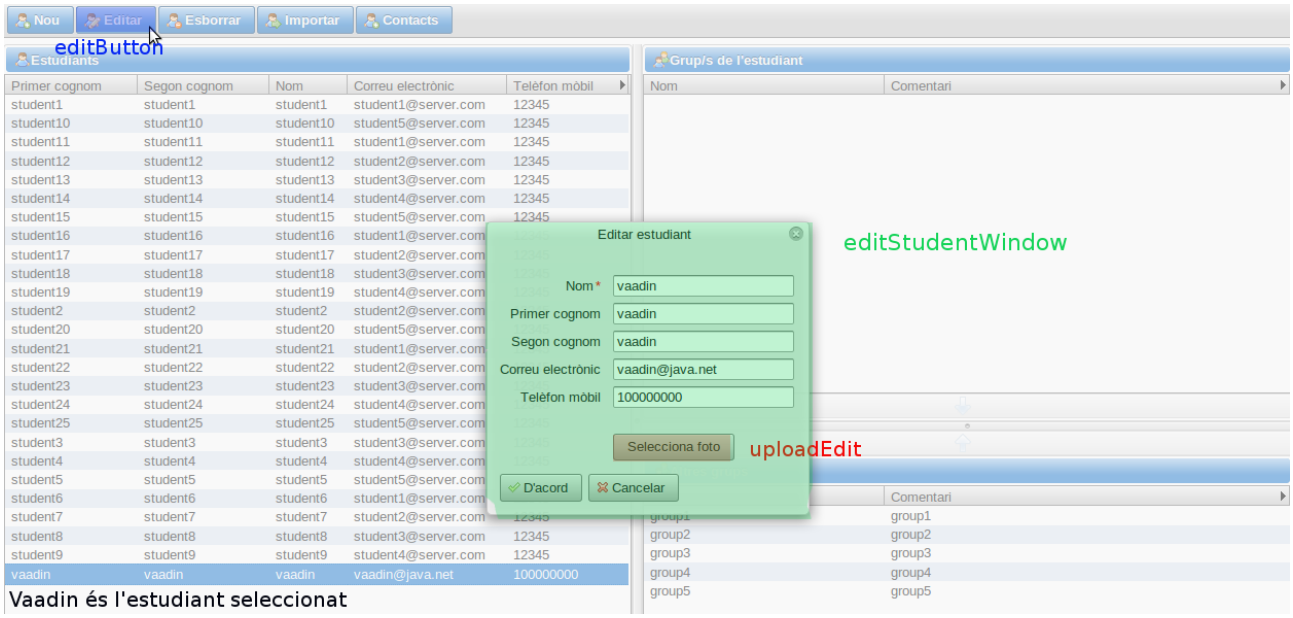


Click a Edita




```
editButton.addListener(new Button.ClickListener() {

        @Override
        public void buttonClick(ClickEvent event) {

            Integer idItem=(Integer) studentsTable.getValue();
            Item item=studentsTable.getItem(idItem);
            Integer idStudent=(Integer) studentsTable.getValue();
            String name=(String) item.getItemProperty("name").getValue();
            String surname1=(String) item.getItemProperty("surname1").getValue();
            String surname2=(String) item.getItemProperty("surname2").getValue();
            String email=(String) item.getItemProperty("email").getValue();
            String smsphone=(String) item.getItemProperty("smsphone").getValue();


            editStudentWindow.setIdStudent(idStudent);
            editStudentWindow.studentForm.getField("name").setValue(name);
            editStudentWindow.studentForm.getField("surname1").setValue(surname1);
            editStudentWindow.studentForm.getField("surname2").setValue(surname2);
            editStudentWindow.studentForm.getField("email").setValue(email);
            editStudentWindow.studentForm.getField("smsphone").setValue(smsphone);
            getWindow().addWindow(editStudentWindow);
        }
    });
```

La finestra que obre l'event anomenada editStudentWindow és la següent:

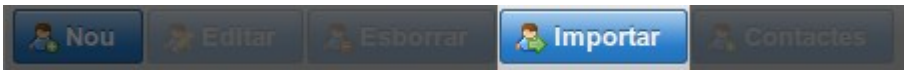


Click a Esborra


```
deleteButton.addListener(new Button.ClickListener() {

        @Override
        public void buttonClick(ClickEvent event) {

            Integer idItem=(Integer) studentsTable.getValue();
            Item item=studentsTable.getItem(idItem);
```

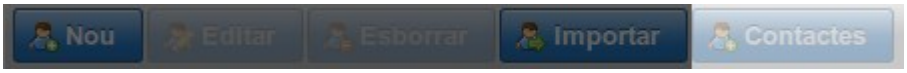
```
                Integer idStudent=(Integer) studentsTable.getValue();
                String name=(String) item.getItemProperty("name").getValue();
                String surname1=(String) item.getItemProperty("surname1").getValue();
                String surname2=(String) item.getItemProperty("surname2").getValue();
                String email=(String) item.getItemProperty("email").getValue();
                String smsphone=(String) item.getItemProperty("smsphone").getValue();;
                deleteStudentWindow.setStudent(new
Student(idStudent,name,surname1,surname2,email,smsphone));
                getWindow().addWindow(deleteStudentWindow);
            }
        });
```

Click a Import



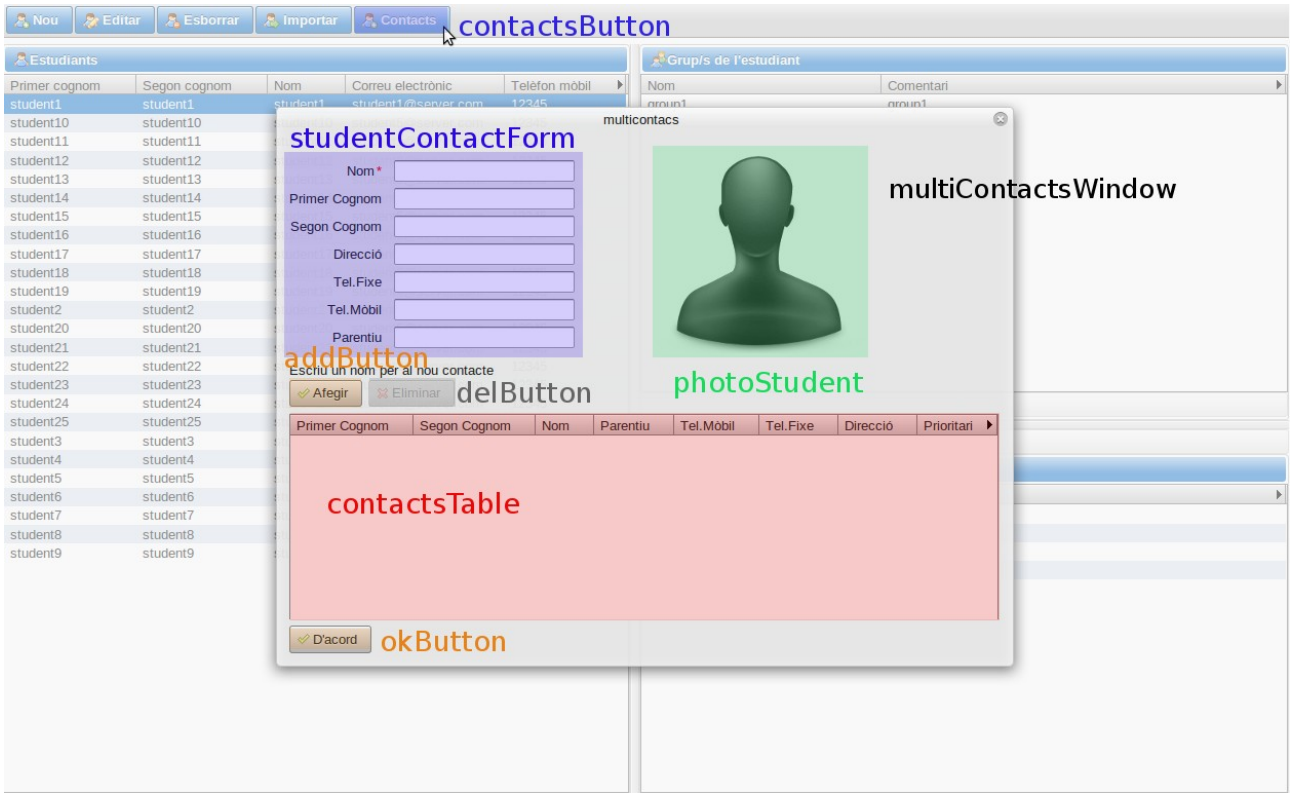

```
importButton.addListener(new Button.ClickListener() {

            @Override
            public void buttonClick(ClickEvent event) {

                getWindow().addWindow(importStudentWindow);
            }
        });
```

Click a Contactes




```
contactsButton.addListener(new Button.ClickListener(){
            @Override
            public void buttonClick(ClickEvent event) {

                Integer idStudent =(Integer) studentsTable.getValue();
                //retornamos lista de conta
                List<Contact> studentContactList = app.storage.getContactCollection(idStudent);

                Student student = app.storage.getStudent(idStudent);

                if(multiContactsWindow.getParent() != null) {
                    // Window already open

                } else {
                    // open window
                    //imagen studient
                    StreamResource imagePhoto = student.getImageResource(getApplication(),
idStudent.toString());
                    if (student.getBytePhotoArray()!=null){
                        photoStudent.setSource(imagePhoto);
                    }else{
                        photoStudent.setSource(new
ClassResource("screenlayouts/logininfo/User.png", getApplication()));
                    }

                    multiContactsWindow.setPhotoStudent(photoStudent);
                    refreshContactsTable(contactsTable, idStudent);
                    getWindow().addWindow(multiContactsWindow);
                }

            }

        });
```

L'event obre una finestra anomenada multiContactsWindow que és la següent:



Métodes implicats a la base de dades

refreshtable()--> Refresca la informació de la tabla especificada.
getStudentsQueryContainer()--> Retorna un contenidor de consulta que consta de tots els estudiants.
getGroupsOfStudentQueryContainer()--> Retorna un contenidor de consulta que consta dels grups de estudiants.
getGroupsHaveNotStudentGroupQueryContainer()--> Retorna un contenidor de consulta que consta dels grups que no tenen estudiants.
addStudent()--> Afegeix un estudiant a la base de dades. Retorna l'identificador (enter) d'aquest.
updateStudent()--> Actualitza la informació de l'estudiant especificat.
deleteStudent()--> Esborra l'estudiant especificat.
addStudentToGroup()--> Afegeix l'estudiant especificat al grup especificat.
deleteStudentFromGroup()--> Esborra un estudiant, especificant el seu grup.
getStudent()--> Obté un estudiant a partir del seu identificador.
addContact()--> Afegeix el contacte a la base de dades.
getContactsQueryContainer()--> Retorna un contenidor de consulta dels contactes especificant l'identificador de l'estudiant.
updateContactPriority()--> Dona prioritat a un contacte a l'hora de enviar notificacions.
contactOfstudent()--> Retorna una col·ecció de contactes de l'estudiant especificat mitjançant el seu identificador.
getContact()--> Retorna un contacte a partir del seu identificador.

Gestió de Assistència

Cas d'ús generic(ABSTRACT)

Resum de la funcionalitat: Cas d'us on un professor pot passar llista, i a més, canviar el seu password..

Paper dins del treball de l'usuari: Aquest cas d'ús és un cas d'ús principal.

Actors relacionats amb el cas d'ús: El professor, el tutor i l'administrador.

Casos d'ús relacionats: Estudiants, Grups, Assignatures, Blocs lectius, Professorat.

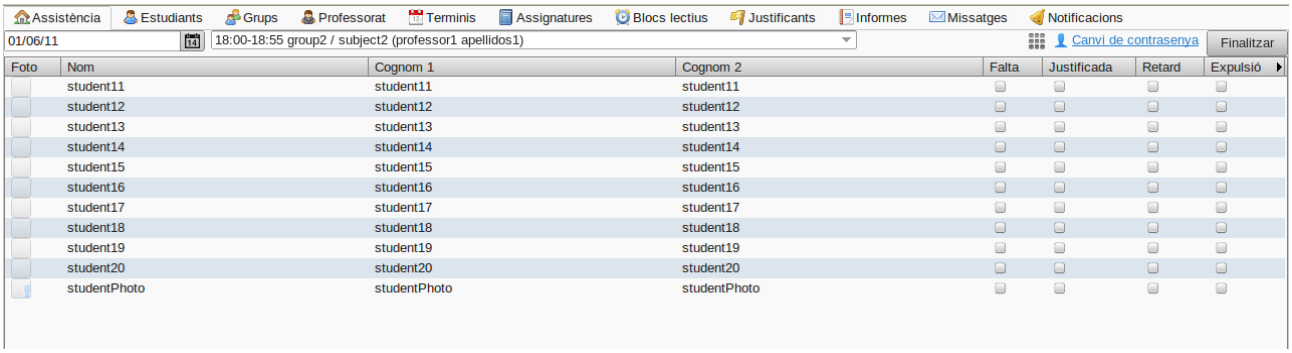
Precondició: Els alumnes tenen que estar a un grup. A més, necessita existir el bloc lectiu.

Postcondició: S'ha passat llista al bloc lectiu seleccionat.

Funcionalitat

En aquest cas d'us, el professor podrà :

1. Gestionar les faltes d'assistència:
   - Mitjançant un calendari, podrà seleccionar el dia a passar llista.
   - En un desplegable podrà seleccionar l'hora de classe que vol segons el dia.
2. També podrà canviar la seva contrasenya d'usuari.



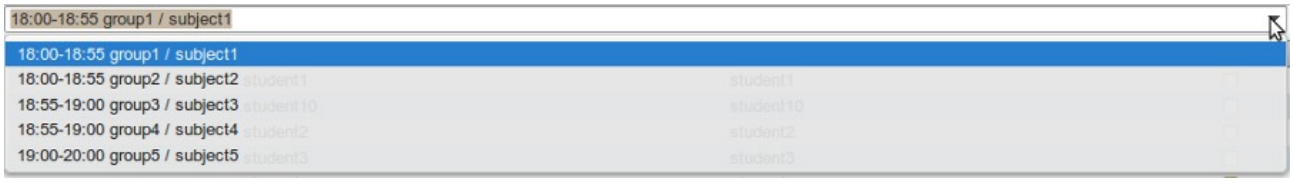
Una vegada el professor ha fet el login a l'AttendaceManager, pot accedir a "Assistència" des-de la barra de menú superior. L'Interfície es divideix en 4 grups.

1. Dia i classe
2. Llistat d'alumnes
3. Canvi de vista
4. Canvi de contrasenya
5. Boto finalitzar

1.Dia i classe.

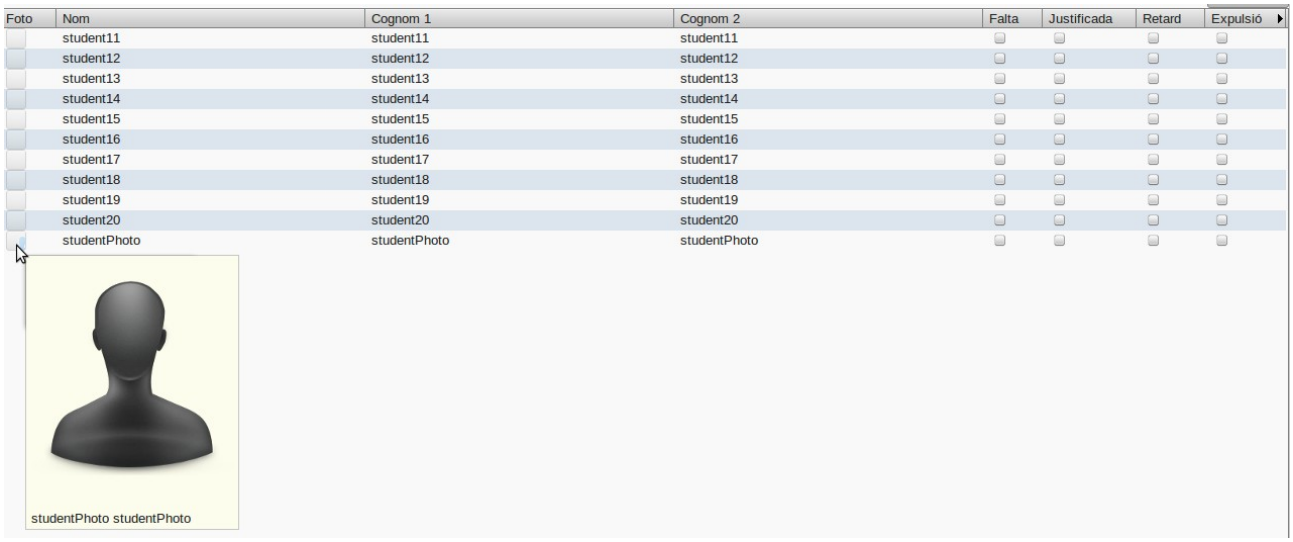
- El professor pot triar el dia que vol gestionar les faltes. Pot posar la data manualment, o pot seleccionarla desde un calendari que s'obre al fer click sobre el boto que hi ha a la dreta.

- El sistema mostra mitjançant un desplegable, totes les classes que te el professor en el dia seleccionat. Despres de fer click a la classe desitjada, pasem a gestionar les assistencies.



2.Llistat d'alumnes

- En aquest apartat, surt una llista dels alumnes de la classe seleccionada. Per cada alumne, es mostra un tooltip amb la foto, el nom, el cognom 1 i el cognom 2, així com els diferents tipus d'estat. Per poder mostrar el tooltip hem de fixar el cursor del ratolí a sobre del camp foto de l'estudiant.



- Es pot ordenar el llistat d'alumnes segons el seu nom o cognoms. A mes, es possible treure o incloure les columnes desitjades, mitjançant la fletxa situada a la dreta del tot de la fila d'ordenació.

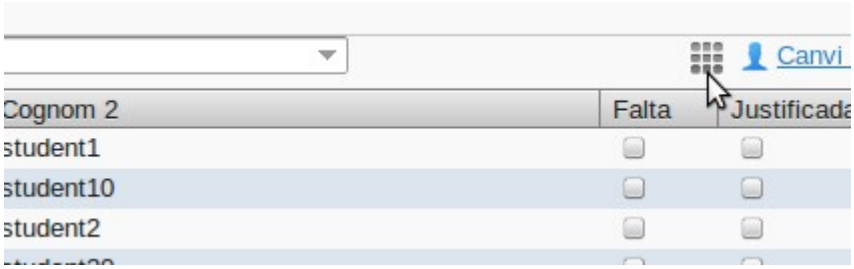
- **Hi han 5 tipus d'estats:**

1. Normal: Es el que te l'alumne abans de que es passi llista. Si esta a classe no es canviara el seu estat.
2. Falta: Quan l'alumne falta a la classe seleccionada.

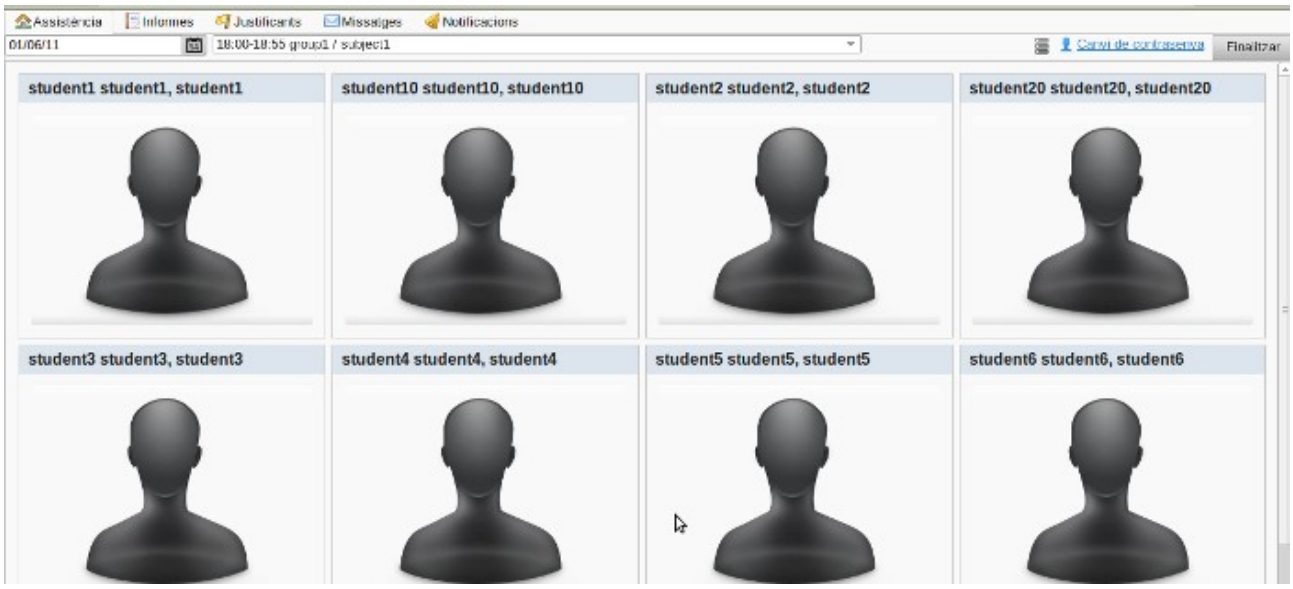
3. Justificada: Quan l'alumne ja ha justificat la seva falta.
4. Retard: Quan l'alumne arriba tard a la classe.
5. Expulsió: Quan l'alumne es expulsat de classe.

3.Canvi de vista

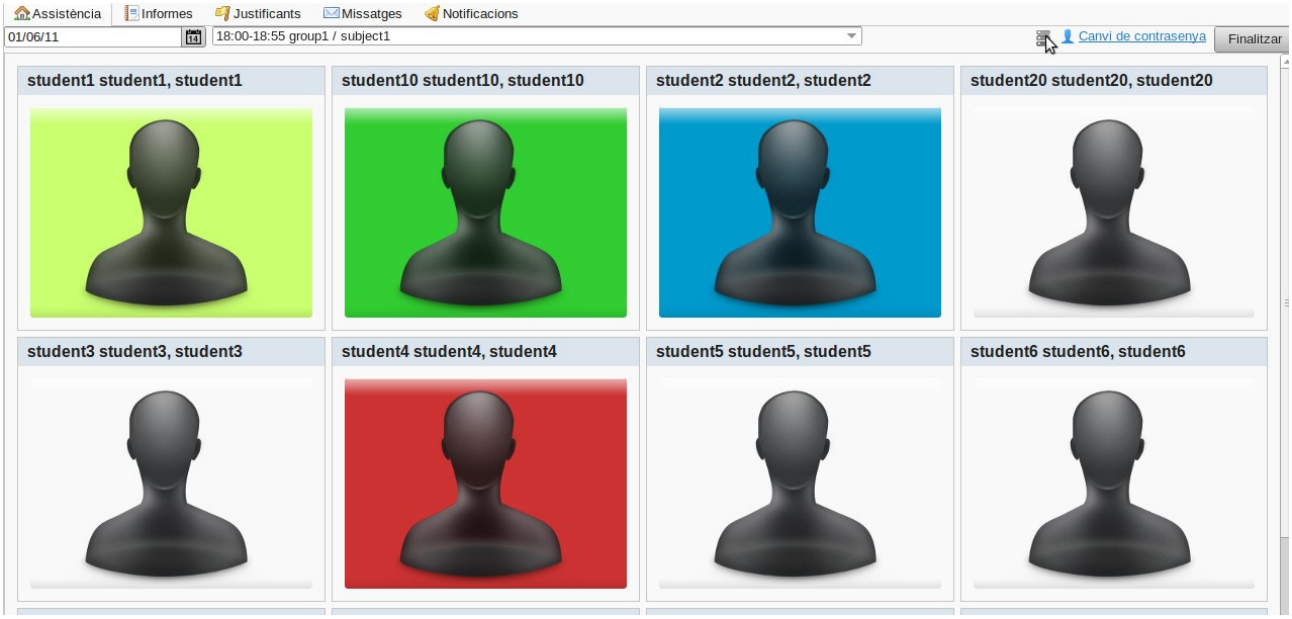
- El professor pot canviar de vista per passar llista mitjançant les fotos dels estudiants.



- La vista que es mostra és la següent:

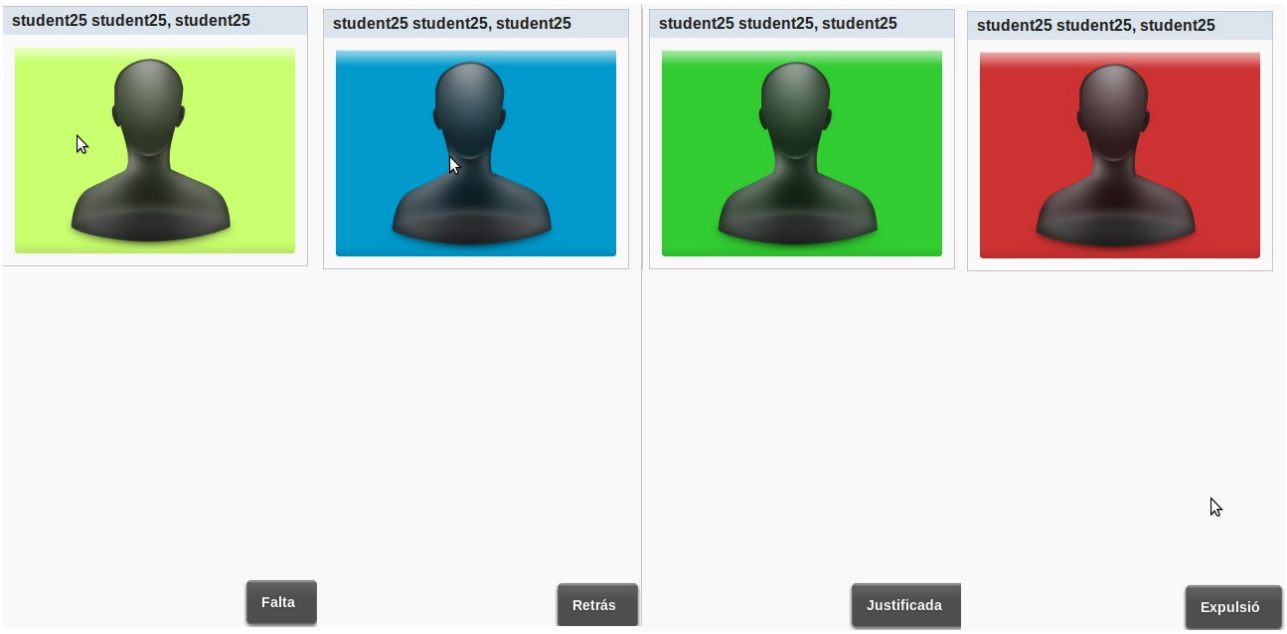


- Per passar llista en aquesta vista hem de clickar sobre la foto de l'estudiant que volem, el fons de la foto canviarà de color segons el tipus de falta que apliquis.

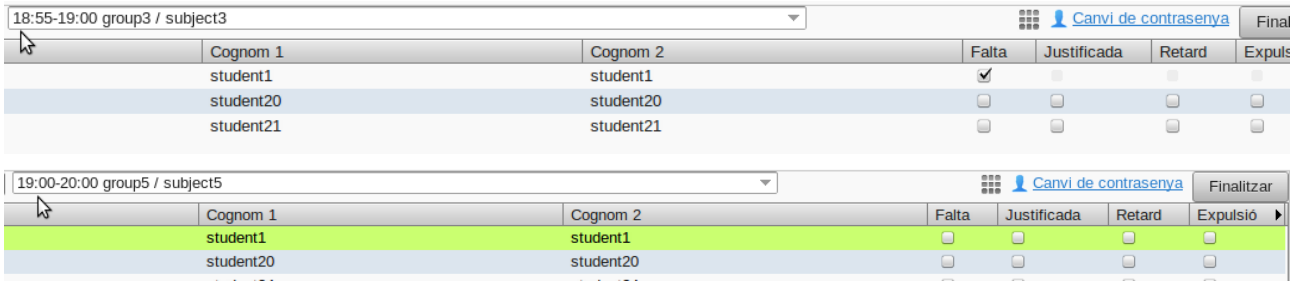


- Al modificar el tipus de falta que té l'aulmne en aquesta vista, simultàneament es modifica a l'altre vista, clickant als checkboxs de la vista en forma de llista també modifica els colors de la vista amb fotos.

- Els tipus de faltes es simbolitzen amb els següents colors:



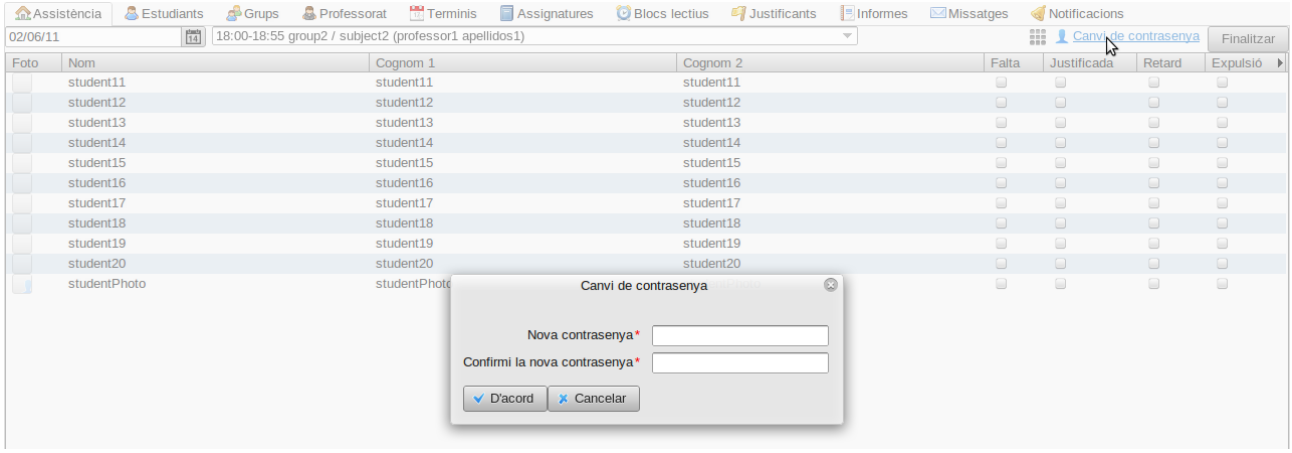
- Quan apliquem falta a un alumne en una hora determinada, ens fixem que a la hora següent, la fila de l'alumne en concret té diferent color segons e ltipus de falta aplicat.



- També s'ha inserit un nou tipus de falta miscel·lànea.

4.Canvi de contrasenya

- El professor podrà canviar la seva contrasenya mitjançant l'enllaç corresponent. S'obrira la següent pantalla per omplir-la i confirmar el canvi.

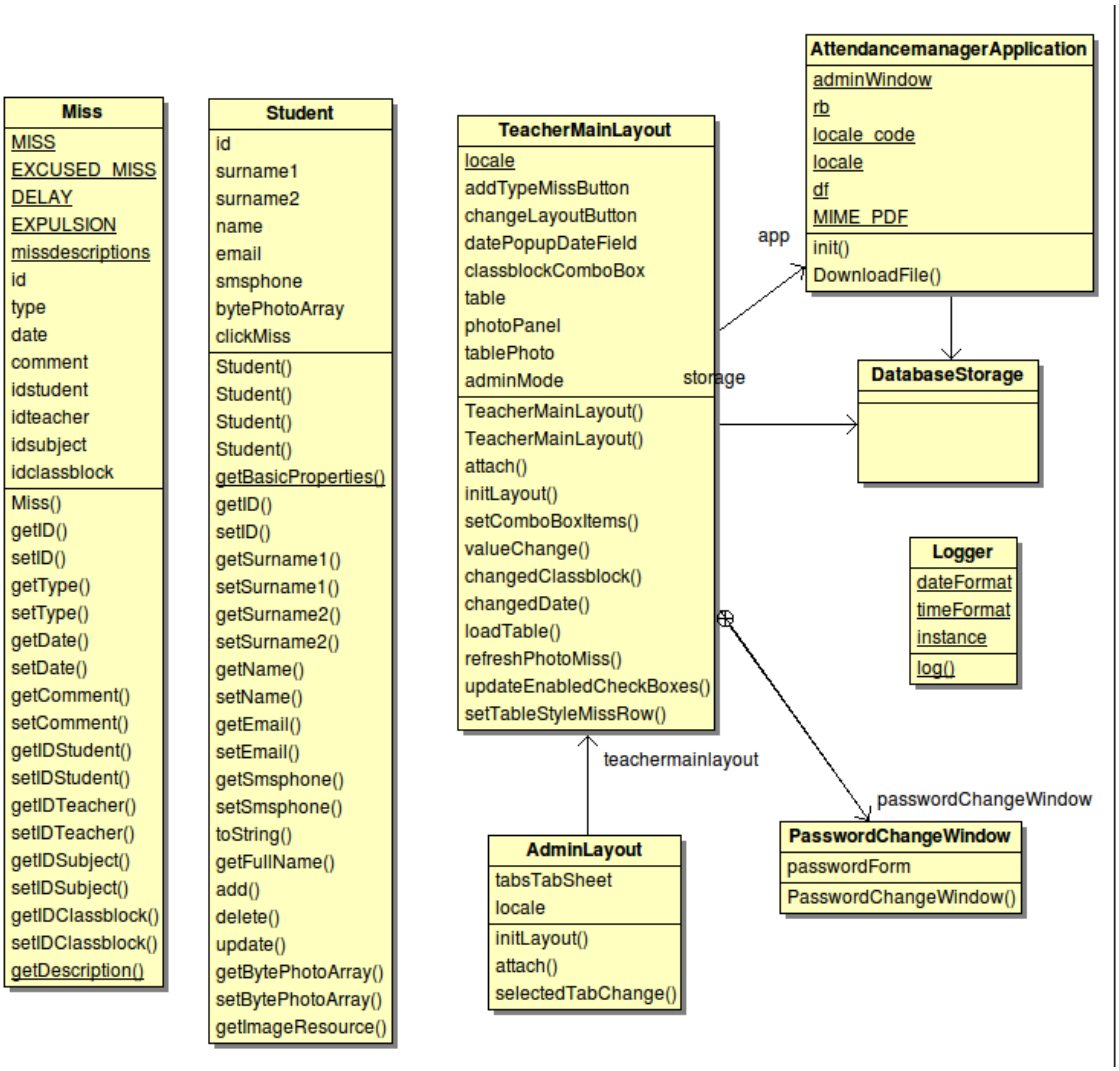


5.Boto finalitzar

- Surt de la aplicacio i torna a la pantalla de login.

Anàlisi

Diagrama de classes implicades al cas d'us



Gestió d'assitència

Diagrama d'actors

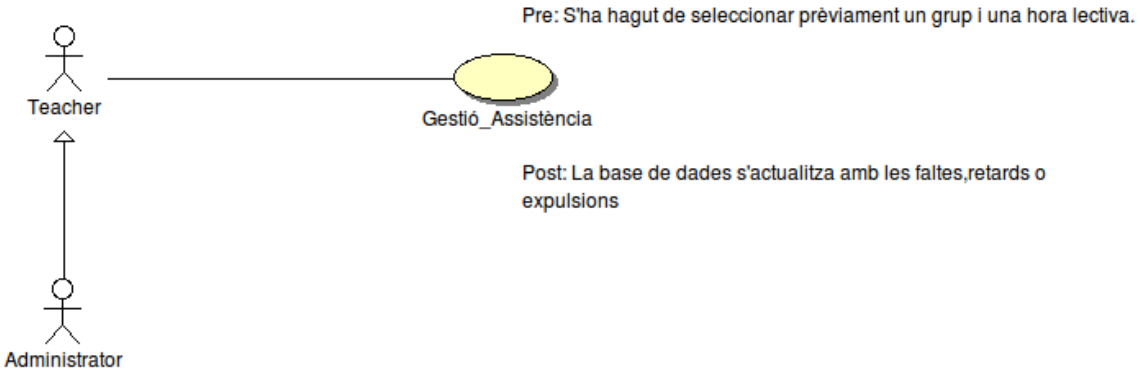
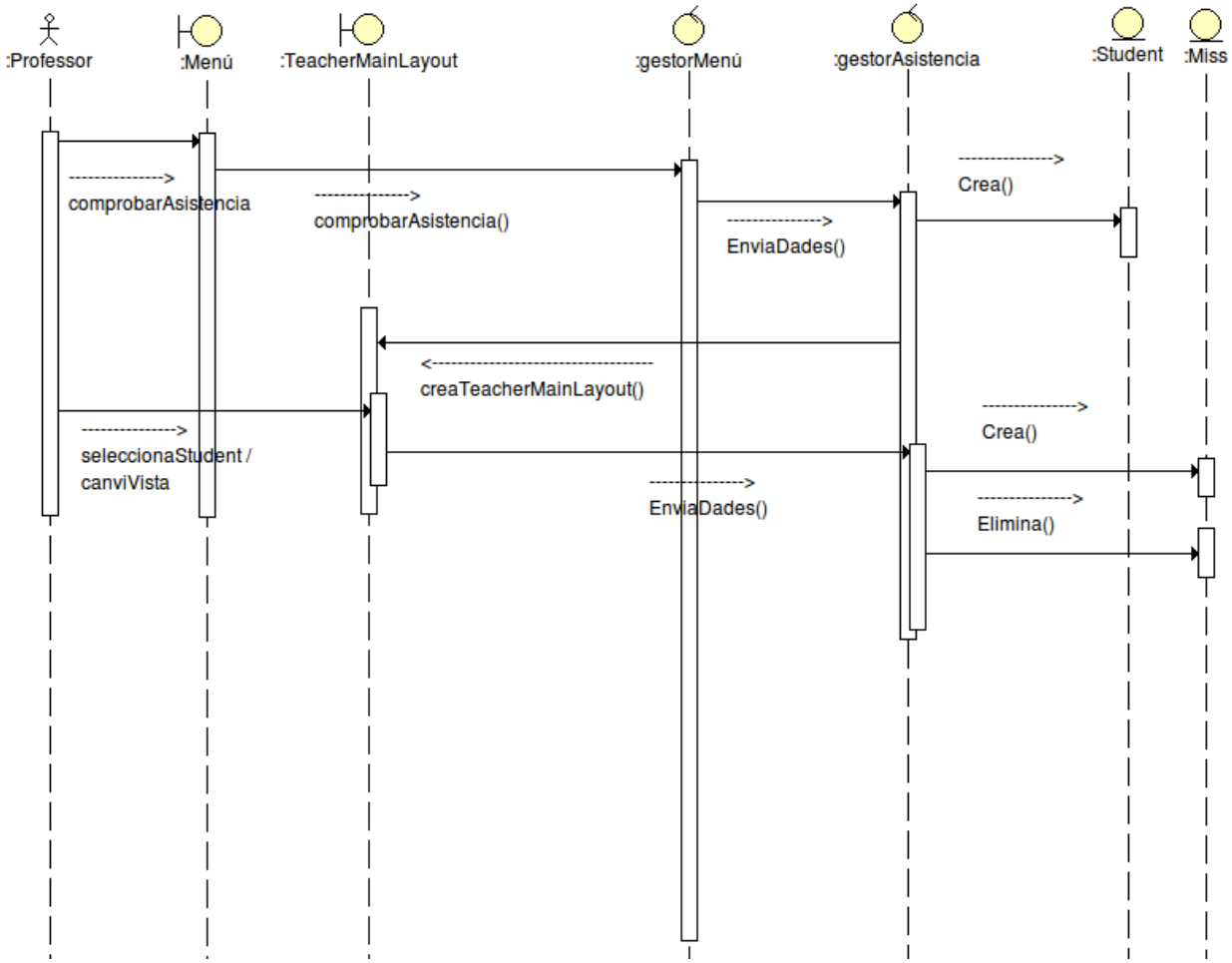


Diagrama de secuencia



- **Classe TeacherMainLayout**

1. **Classe PasswordChangeWindow**

2. **Classe PasswordInfo**
3. **Classe AttendanceManagerApplication**
4. **Classe DatabaseStorage**
5. **Classes d'entitat**
    - *Classe Logger*
    - *Classe Miss*
    - *Classe Student*

Metodes de la classe

public TeacherMainLayout()

- Aquest es el mètode constructor per defecte. Hi han dos tipus. Sense cap argument, crea l'objecte com a professor normal. Si es crida al mètode amb un paràmetre de tipus boolean, aquest creara l'objecte per un usuari administrador.

public void attach()

- Aquest mètode s'encarrega d'inicialitzar el layout i els seus classblock segons el tipus d'usuari que hagi fet el login.

public void initLayout()

- Aquest metode s'encarrega de dibuixar el layout amb els seus components.

private void setComboBoxItems(ComboBox combo, Collection<?> collection)

- Aquest metode s'encarrega de omplir el ComboBox.

public void valueChange(ValueChangeEvent event)

- Aquest metode es el que canvia els estats dels alumnes quan hi ha un canvi.

public void changedClassblock(ValueChangeEvent event)

- Aquest metode s'encarrega de canviar els alumnes de la llista quan el professor fa un canvi de grup.

public void changedDate(ValueChangeEvent event)

- Aquest metode s'encarrega de canviar els grups de la llista quan el professor fa un canvi en el calendari.

public void loadTable(int idgroup)

- Aquest metode s'encarrega de carregar totes les dades referents a un grup enmagatzemadas a la BBDD.

private void updateEnabledCheckBoxes(Student student)

- Aquest metode s'encarrega de actualitzar els estats dels checkbox que estan marcats.

private void refreshPhotoMiss(Student student, Button studentButton)

- Métode que canvia el color de fons de les fotos segons el tipus de falta que té l'estudiant.

public void setTableStyleMissRow()

- Métode que atribueix color a les files de la taula d'estudiants si aquests tenen faltes a les hores anteriors.

Classes aniuades

Classe PasswordChangeWindow

```
private class PasswordChangeWindow extends Window {
```

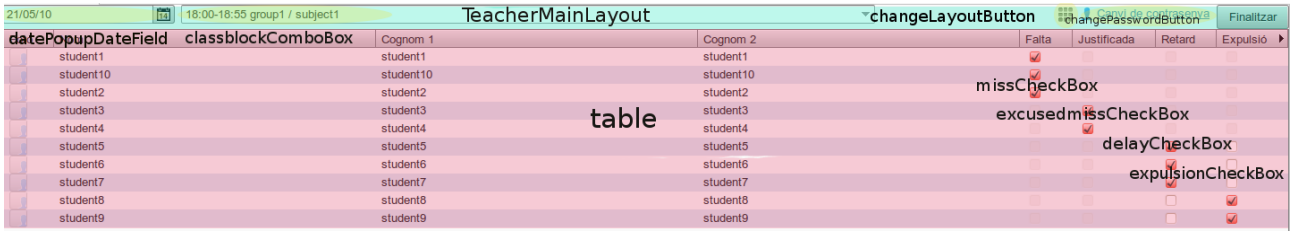
- Aquesta classe defineix i gestiona la finestra per fer el canvi del password del professor.

Classe PasswordInfo

```
public class PasswordInfo {
```

- Aquesta classe aniuada dins de la classe ***PasswordChangeWindow***, retorna la informació dels passwords dels usuaris.

Disseny



TeacherMainLayout

optionsLayout

- datePopupDateField
- classblockComboBox
- changeLayoutButton
- changePasswordButton
- logoutButton

table

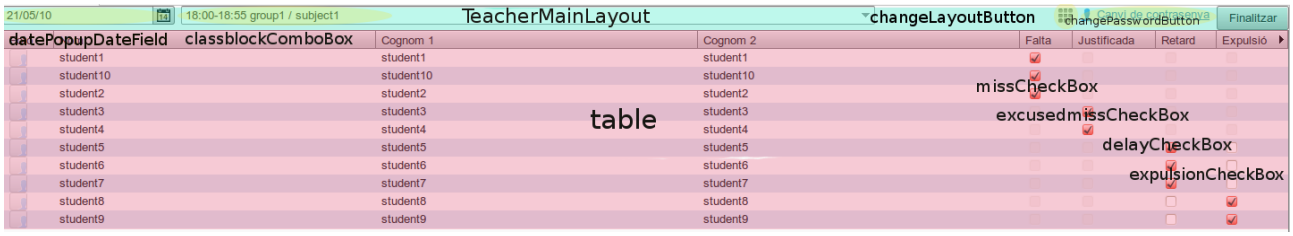
- missCheckBox
- excusedmissCheckBox
- delayCheckBox
- expulsionCheckBox

photoPanel

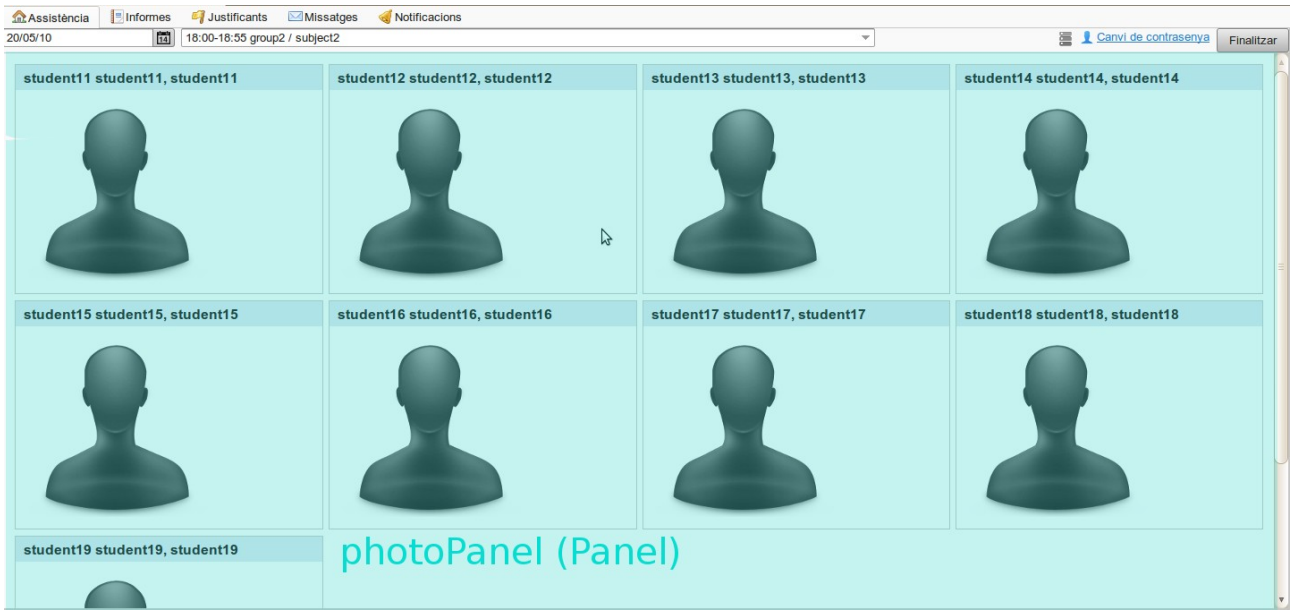
- tablePhoto
  - studentBox
    - studentButton

* **optionsLayout**

* **table**

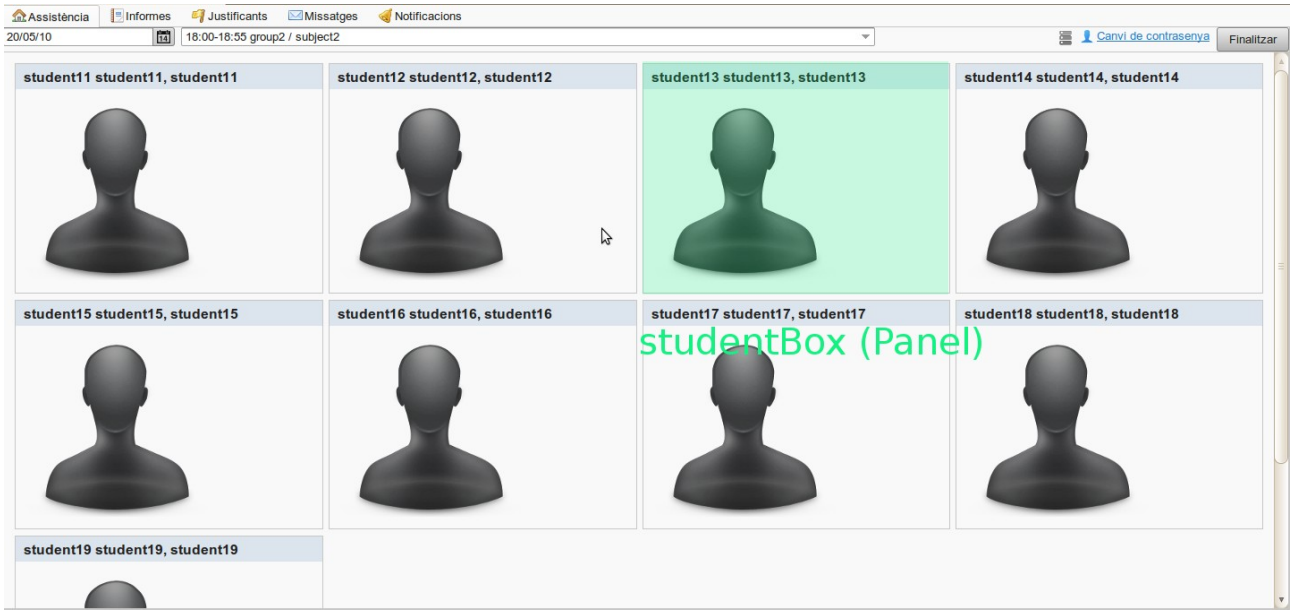


* **photoPanel**

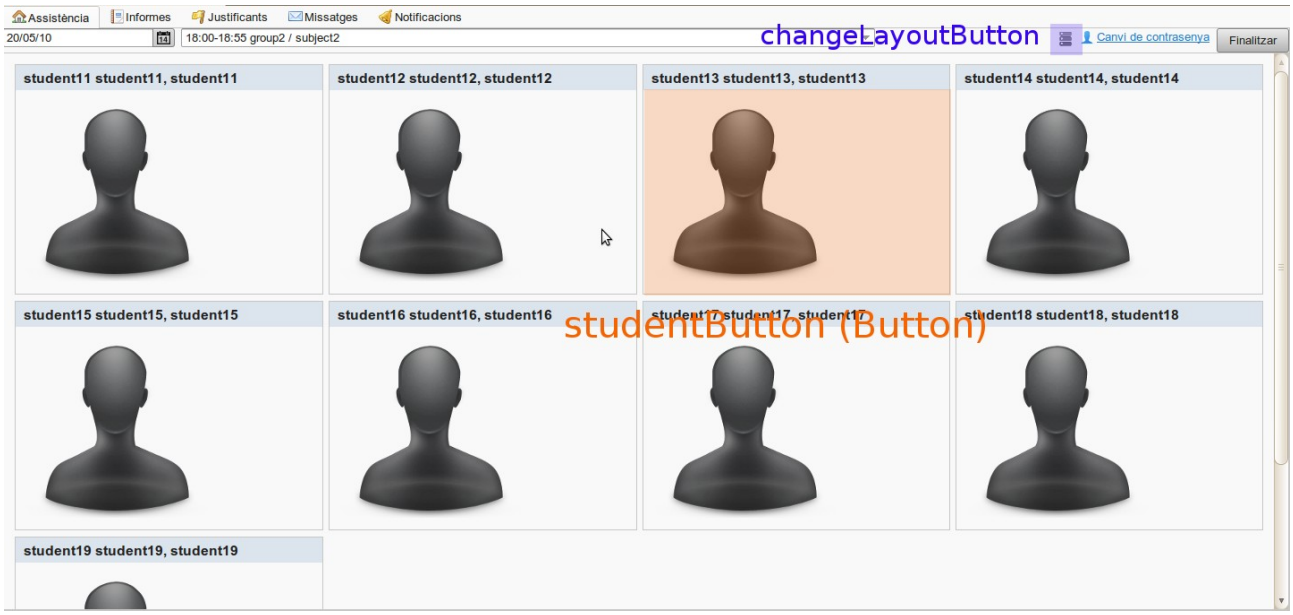


* **tablePhoto**

*** studentBox**



* **studentButton**



Aspectes remarcables de la implementació

Click a changeLayout

```
changeLayoutButton.addListener(new ClickListener(){
        @Override
        public void buttonClick(ClickEvent event) {

                int indexOfTable = getComponentIndex(table);
                if (indexOfTable!=-1){
                        // TABLA DE FOTOS
                        //redresco de la tabla
                        if (classblock!=null){
```

```
                                        loadTable(classblock.getIDGroup());
                                }
                                removeComponent(table);
                                addComponent(photoPanel);
                                setExpandRatio(photoPanel, 1);
                                setSizeFull();
                                changeLayoutButton.setIcon(new
ClassResource("screenlayouts/logininfo/list.png", getApplication()));
                        }
                        else{
                                //TABLA NORMAL
                                //redresco de la tabla
                                if (classblock!=null){
                                        loadTable(classblock.getIDGroup());
                                }
                                removeComponent(photoPanel);
                                addComponent(table);
                                setExpandRatio(table, 1);
                                setSizeFull();
                                changeLayoutButton.setIcon(new
ClassResource("screenlayouts/logininfo/grid.png", getApplication()));
                        }
                }
        });
```

click a studentButton

```
                studentButton.addListener(new ClickListener(){
                        @Override
                        public void buttonClick(ClickEvent event) {
                                refreshPhotoMiss(student,studentButton);
                                tablePhoto.requestRepaint();
                        }
                });
```

Métodes implicats a la base de dades

studentsOfGroup()--> modificació del métode per poder afegir el camp photoid de l'estudiant. Recull tots els atributs d'un estudiant i crea un objecte que afegeix a una col·lecció que posteriorment és retornada

classblocksOnDate(new Date())--> métode que retorna una col·lecció de blocs lectius.

classblocksOfTeacherOnDate(app.user.getID(), new Date())--> métode que retorna una col·lecció de blocs lectius d'un professor en concret.

studentsOfGroup(idgroup)--> métode que retorna una col·lecció d'estudiants d'un grup en concret.

existsMiss(Miss.MISS, (Date)datePopupDateField.getValue(), ((Classblock)classblockComboBox.getValue()).getID(), student.getID())--> métode que retorna un boleà, true si l'alumne en un bloc lectiu en concret i un tipus de falta, false el contrari.

addMiss(miss)--> retorna l'identificador de la falta.

deleteMiss(Miss.MISS, miss.getDate(), miss.getIDStudent(), miss.getIDClassblock())--> elimina una falta d'un alumne en un bloc lectiu i data especificats.

getPasswordHash(passwordInfo.passwordA)--> retorna una string del password en forma hexadecimal.

updateTeacher(app.user)--> actualitza un professor.

Gestió de notificacions

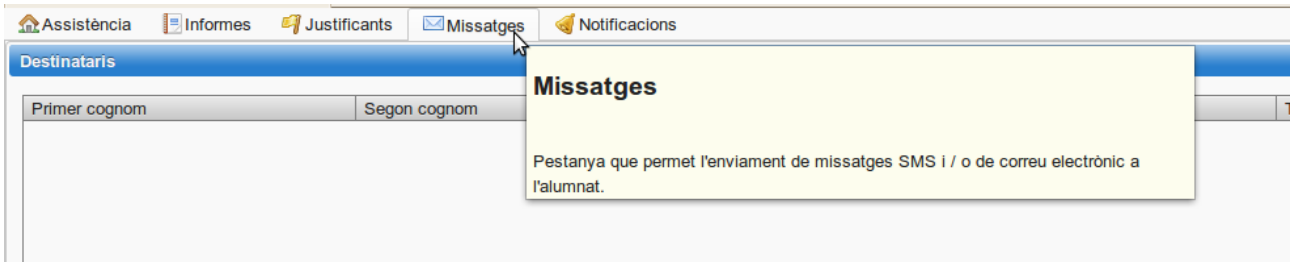
Funcionalitat

Accés

- El professor es logueja amb el ser usuari i passwords enregistrats a la base de dades
- (previament ha de estar donat d'alta per un usuari amb privilegis Admin).
- Accedeix al apartat de “Missatges” per tal de enviar les notificacions necesaries via sms/email als alumnes.

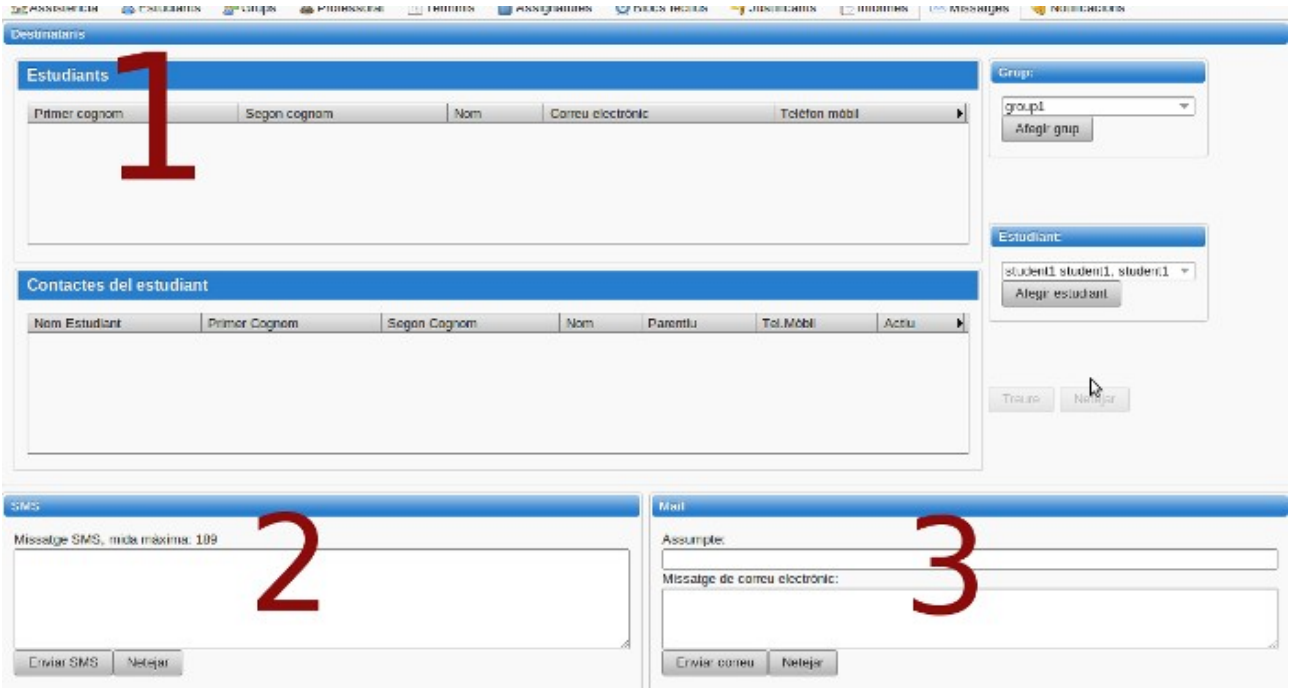
Cas d’ús no 1 : *Gestió Missatgeria* (Abstracta)

- Resum de la funcionalitat: Enviar missatges via sms e-mail al alumnat
- Paper dins del treball de l’usuari: aquest cas d’ús és un dels casos del professor o tutor
- Actors relacionats amb el cas d’ús: Professor
- Casos d’ús relacionats: Alta estudiants
- Precondició: la base de dades ha de tenir estudiants en el grup del professor
- Postcondició: missatje enviat a pares o tutors de l'alumne
- Descripció del cas d’ús:
- Comuna: El professorat accedeix al apartat de missategeria, selecciona el grup associat amb aquet professor i afegeix els alumnes que vol enviar missatges als seus pares o tutors corresponents.



Vista

- L'usuari es troba amb la pantalla de missatgeria on hi han 3 apartats diferents:

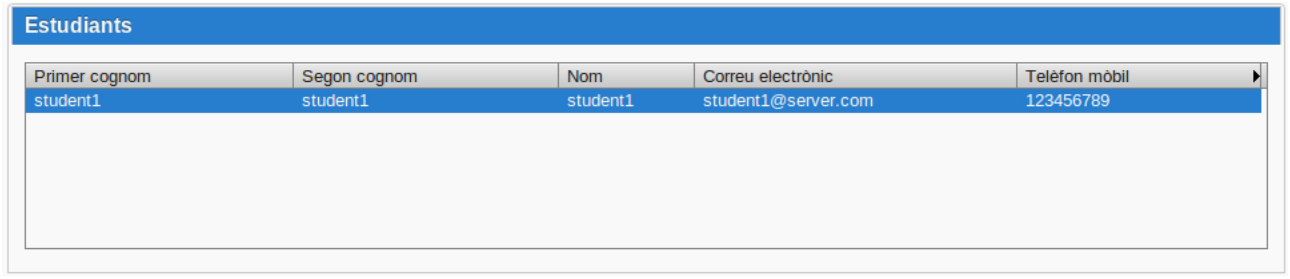


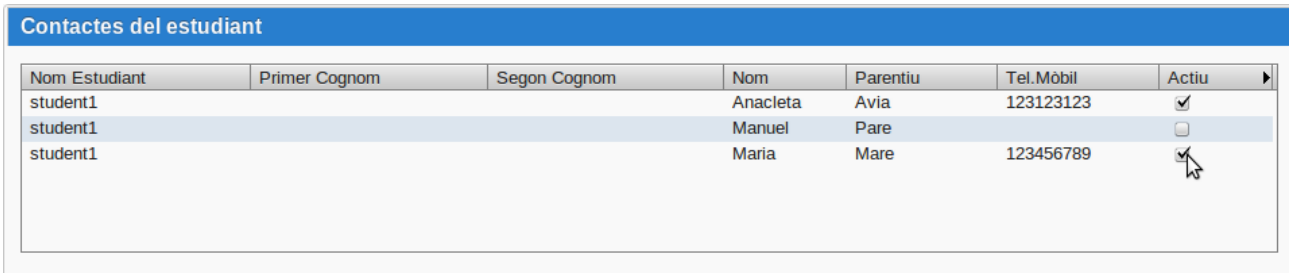
Afegir alumnat

- El primer apartat permet afegir alumnat a una llista, pot afegir un grup que pertanyi, de cop, o per separat.
- Aquesta llista es pot modificar l'alumnat , eliminat de la llista o netejant la llista sencera.
- Unicament deixarem l'alumnat que volem que rebi missatgeria de notificacio.
- A la taula de Contactes dels estudian, automaticament s'insertaran els contactes d'aquet estudiant, per tal de visualitzar-los,

Els contactes amb “Actiu” seleccionat serán els que els arribi els missatges

- (l'alumnat de la llista conté informacio de numero de mobil de tutor i correu)

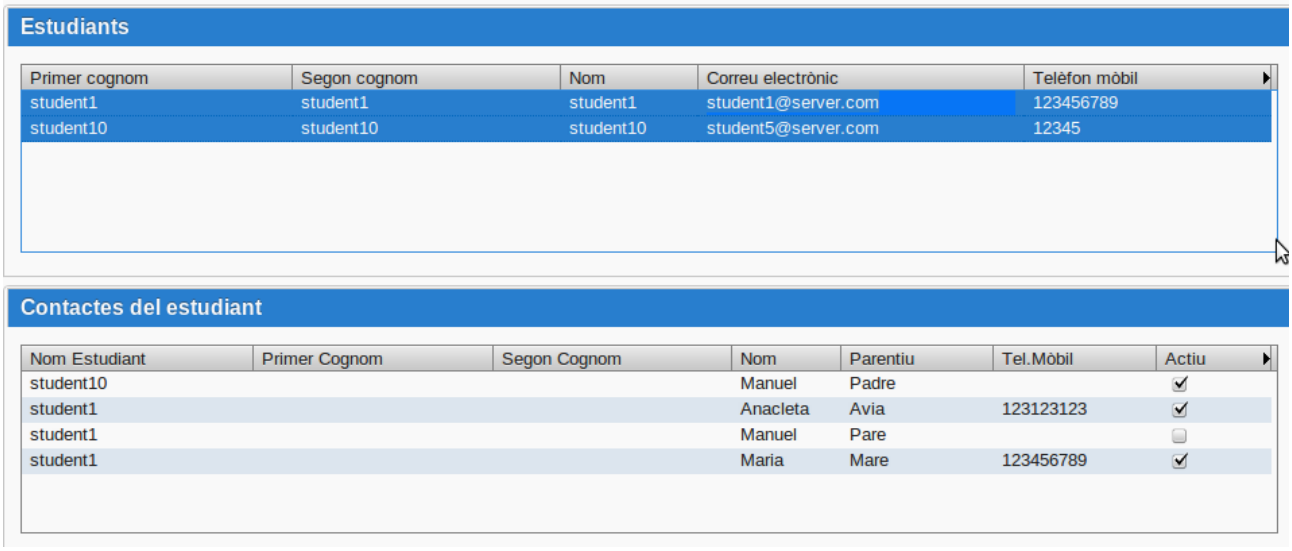




Multiselecció

- A l'hora d'afegir m'es d'un alumne per tal de envia missatgeria automaticament es mostrarán tots els contactes de tots aquet's seleccionats

i si son actius o no, es pot modificar aquesta opció en tem's d'execució del programa.

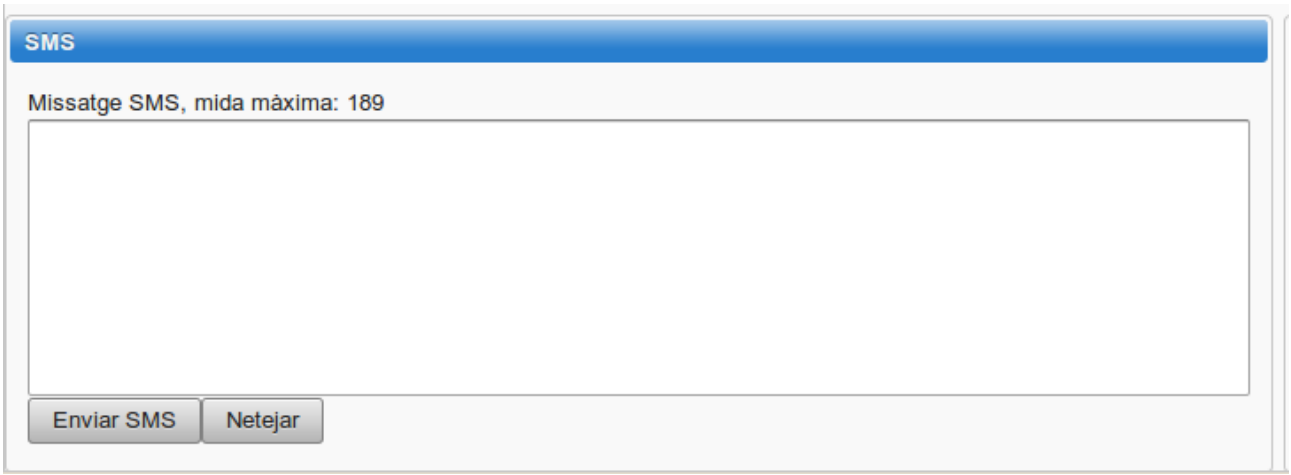


Nou missatge SMS

- El seguient apartat permet al usuario introduir el missatge de text per se enviat via SMS als alumnes seleccionats a la llista.

Cas d'ús no 2 : *Nou SMS* (Concreta)

- Resum de la funcionalitat: Enviar missatges via sms als pares/tutors de l'alumnat
- Paper dins del treball de l'usuari: aquest cas d'ús és un dels casos del professor o tutor
- Actors relacionats amb el cas d'ús: Professor
- Casos d'ús relacionats: Alta estudiants , Gestió Missatgeria
- Precondició: la base de dades ha de tenir estudiants en el grup del professor, i estar previament selecionats els alumnes al cas d'us anterior.
- Postcondició: sms enviat a pares o tutors de l'alumne
- Descripció del cas d'ús:
- Comuna: El professorat accedeix al apartat de missategeria, selecciona el grup associat amb aquet professor i afegeix els alumnes que vol enviar sms als seus pares o tutors corresponents.

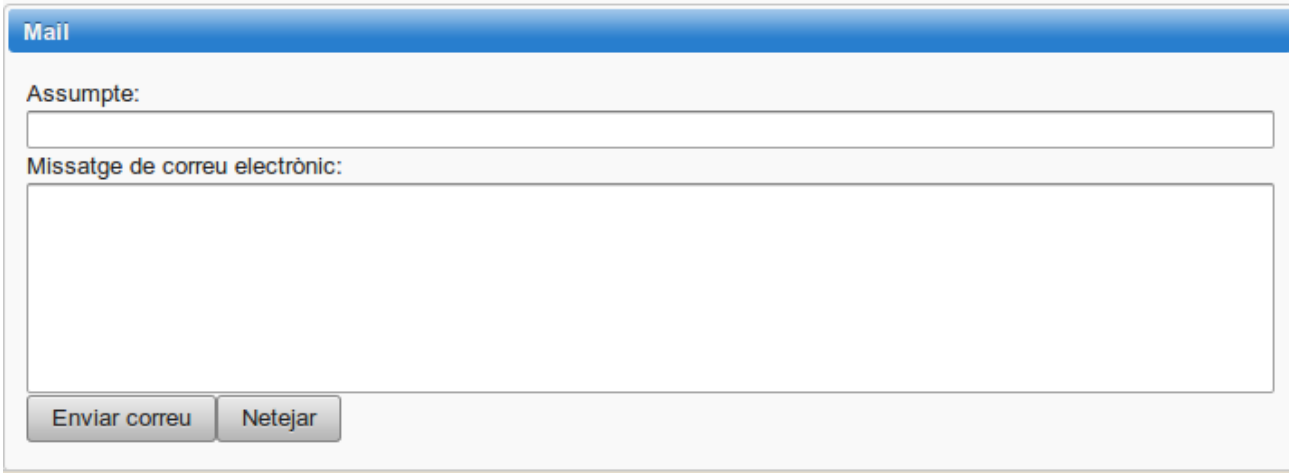


Nou missatge Correu Electronic

- Per ultim tenim el apartat de e-Mail que ens permet introduir el missatge de text y repetir l'operacio pero amb la direccio de correu electronic.

Cas d'ús no 3 : *Nou Email* (Concreta)

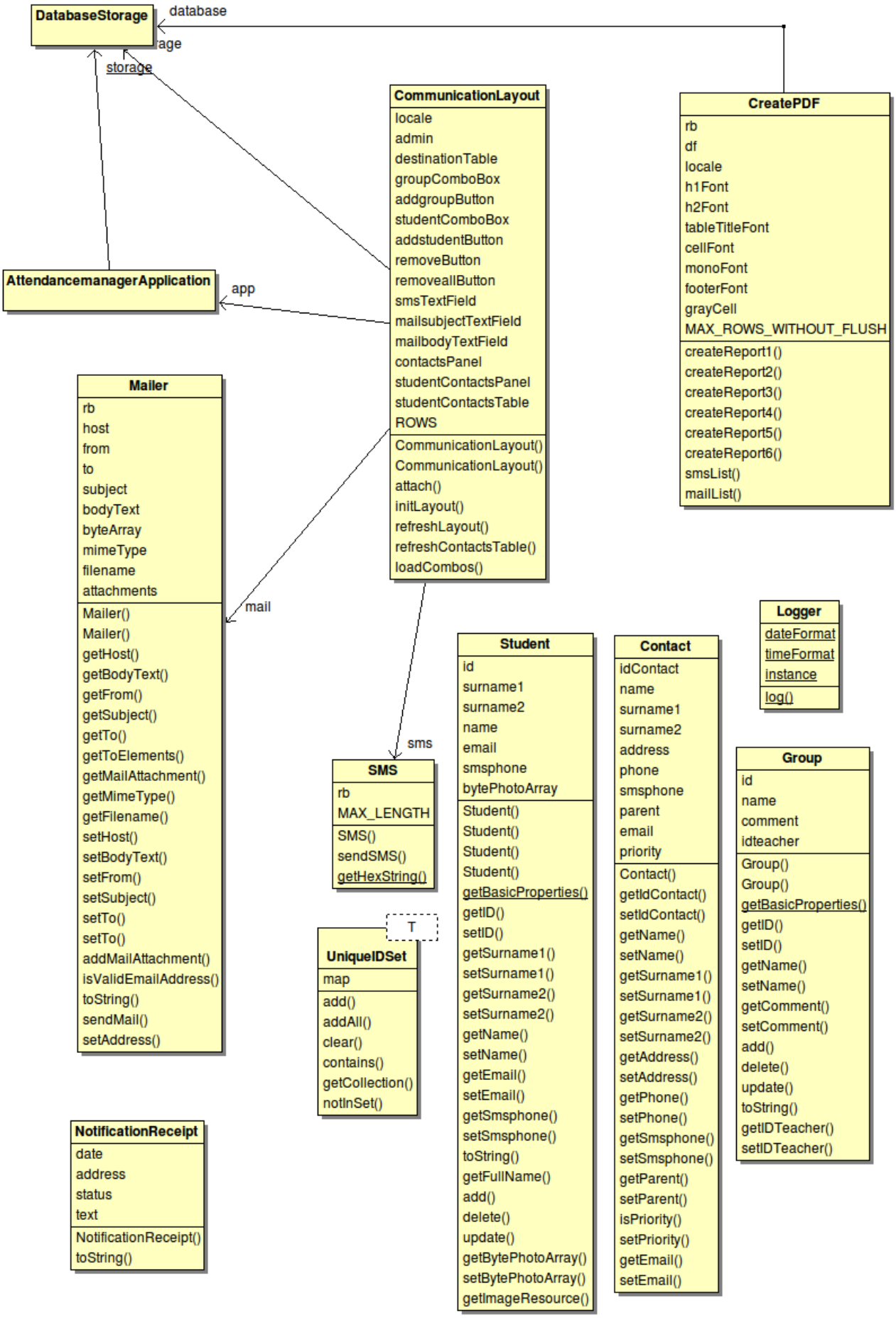
- Resum de la funcionalitat: Enviar missatges via e-mail als pares/tutors de l' alumnat
- Paper dins del treball de l'usuari: aquest cas d'ús és un dels casos del professor o tutor
- Actors relacionats amb el cas d'ús: Professor
- Casos d'ús relacionats: Alta estudiants , Gestió Missatgeria
- Precondició: la base de dades ha de tenir estudiants en el grup del professor, i estar previament selecionats els alumnes al cas d'us anterior.
- Postcondició: email enviat a pares o tutors de l'alumne
- Descripció del cas d'ús:
- Comuna: El professorat accedeix al apartat de missategeria, selecciona el grup associat amb aquet professor i afegeix els alumnes que vol enviar emails als seus pares o tutors corresponents.



Anàlisi

Diagrama de classes implicades al cas d'us

- DatabaseStorage
- AttendancemanagerApplication
- CommunicacionLayout
- Group
- SMS
- Mailer
- UniqueIDSet
- NotificationReceipt
- Student
- Contact
- Logger
- CreatePDF


DatabaseStorage
database
storage
CommunicationLayout
locale
admin
destinationTable
groupComboBox
addgroupButton
studentComboBox
addstudentButton
removeButton
removeallButton
smsTextField
mailsubjectTextField
mailbodyTextField
contactsPanel
studentContactsPanel
studentContactsTable
ROWS
CommunicationLayout()
CommunicationLayout()
attach()
initLayout()
refreshLayout()
refreshContactsTable()
loadCombos()
CreatePDF
rb
df
locale
h1Font
h2Font
tableTitleFont
cellFont
monoFont
footerFont
grayCell
MAX_ROWS_WITHOUT_FLUSH
createReport1()
createReport2()
createReport3()
createReport4()
createReport5()
createReport6()
smsList()
mailList()
AttendancemanagerApplication
app
Mailer
rb
host
from
to
subject
bodyText
byteArray
mimeType
filename
attachments
Mailer()
Mailer()
getHost()
getBodyText()
getFrom()
getSubject()
getTo()
getToElements()
getMailAttachment()
getMimeType()
getFilename()
setHost()
setBodyText()
setFrom()
setSubject()
setTo()
setTo()
addMailAttachment()
isValidEmailAddress()
toString()
sendMail()
setAddress()
mail
sms
SMS
rb
MAX_LENGTH
SMS()
sendSMS()
getHexString()
T
UniqueIDSet
map
add()
addAll()
clear()
contains()
getCollection()
notInSet()
Student
id
surname1
surname2
name
email
smsphone
bytePhotoArray
Student()
Student()
Student()
Student()
getBasicProperties()
getID()
setID()
getSurname1()
setSurname1()
getSurname2()
setSurname2()
getName()
setName()
getEmail()
setEmail()
getSmsphone()
setSmsphone()
toString()
getFullName()
add()
delete()
update()
getBytePhotoArray()
setBytePhotoArray()
getImageResource()
Contact
idContact
name
surname1
surname2
address
phone
smsphone
parent
email
priority
Contact()
getIdContact()
setIdContact()
getName()
setName()
getSurname1()
setSurname1()
getSurname2()
setSurname2()
getAddress()
setAddress()
getPhone()
setPhone()
getSmsphone()
setSmsphone()
getParent()
setParent()
isPriority()
setPriority()
getEmail()
setEmail()
Logger
dateFormat
timeFormat
instance
log()
Group
id
name
comment
idteacher
Group()
Group()
getBasicProperties()
getID()
setID()
getName()
setName()
getComment()
setComment()
add()
delete()
update()
toString()
getIDTeacher()
setIDTeacher()
NotificationReceipt
date
address
status
text
NotificationReceipt()
toString()

Gestió de notificacions

Diagrama d'actors

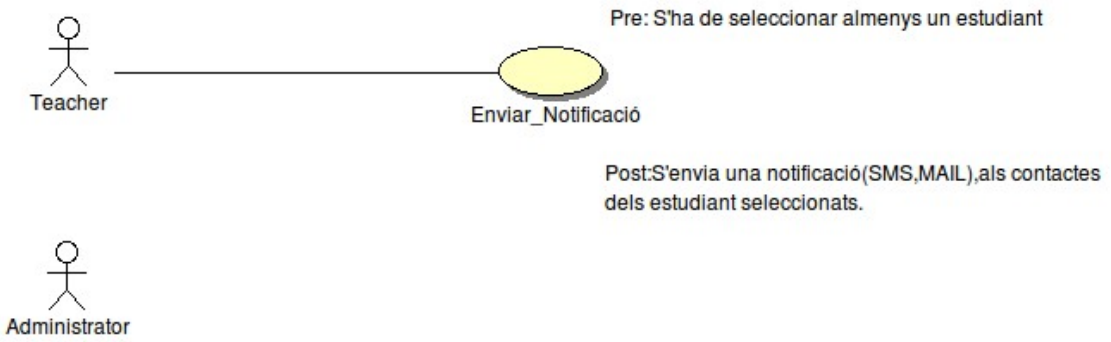
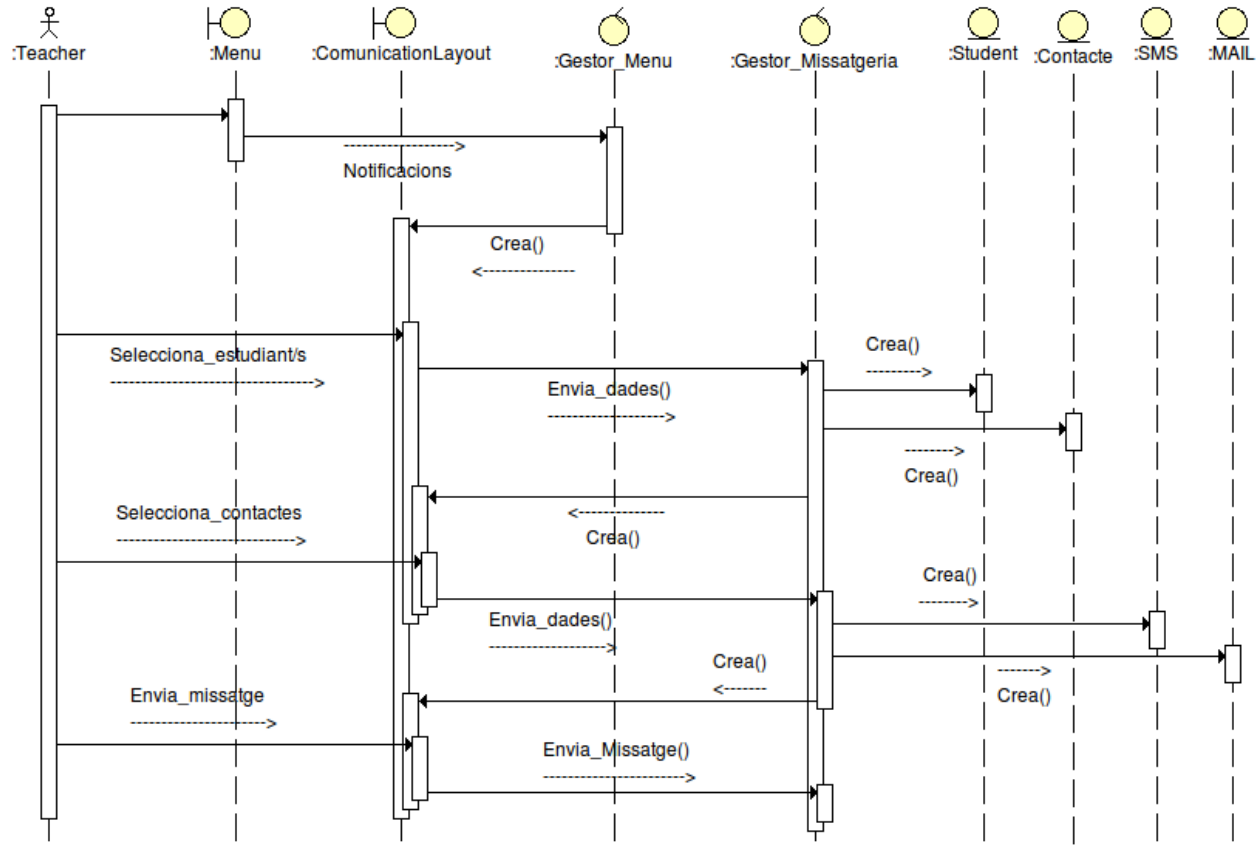


Diagrama de secuencia



Metodes de la classe

public void attach()

- Aquest mètode s'encarrega d'inicialitzar el layout i els seus classblock segons el tipus d'usuari que hagi fet el login.

public void initLayout()

- Aquest metode s'encarrega de dibuixar el layout amb els seus components.

private void refreshContactsTable(Table contactsTable, Student student, boolean multiselect)

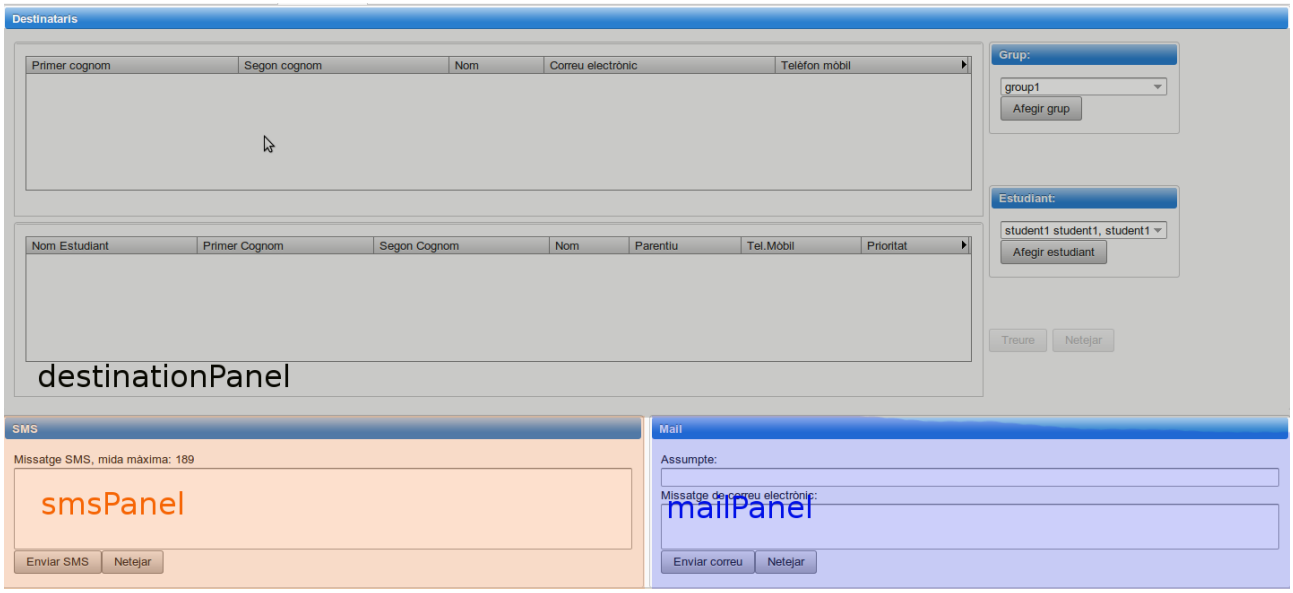
- Refresca la taula de contactes (studentContactsTable).

public void loadCombos()

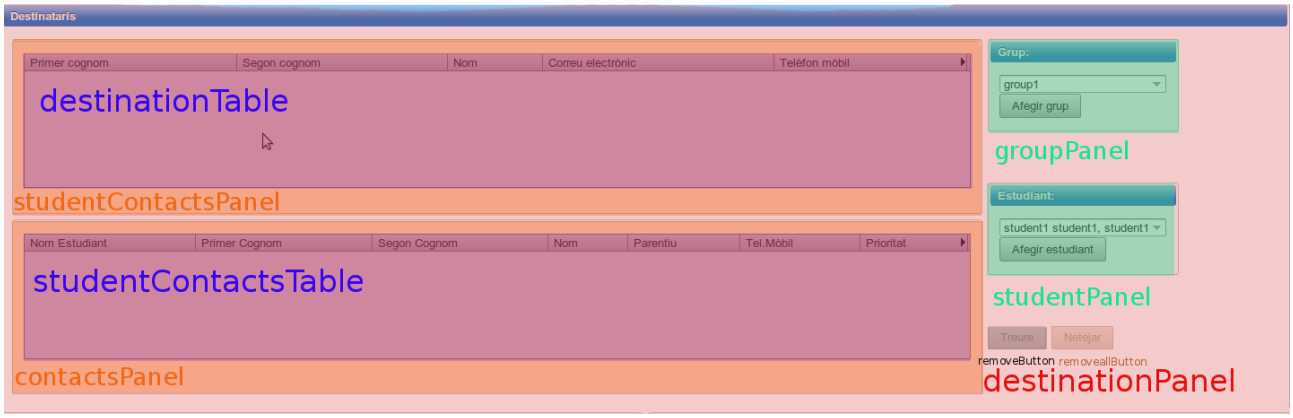
- Refresca els comboBox que componen el layout.

Disseny

ComunicationLayout (verticalLayout)



- **destinationPanel**
  - studentContactsPanel
    - destinationTable
  - contactsPanel
    - studentContactsTable
  - groupPanel
  - studentPanel
  - removeButton
  - removeallButton

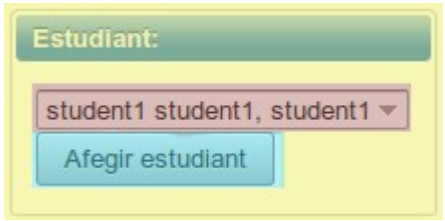


- **groupPanel**
  - COMMUNICATIONLAYOUT_GROUPPANEL_CAPTION (“Grup”)

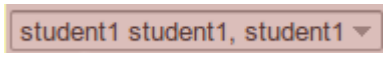
- gourpComboBox
  - COMMUNICATIONLAYOUT_SELECTGROUP_CAPTION (“Seleccioni un grup”)

- addGroupButton
  - COMMUNICATIONLAYOUT_ADDGROUP_CAPTION (“Afegir grup”)

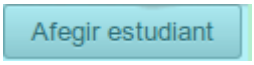
- **studentPanel**
  - COMMUNICATIONLAYOUT_ADDSTUDENT_CAPTION
  - COMMUNICATIONLAYOUT_STUDENTPANEL_CAPTION (“Estudiant”)



- studentComboBox
  - COMMUNICATIONLAYOUT_SELECTSTUDENT_CAPTION (“Seleccioni un estudiant”)

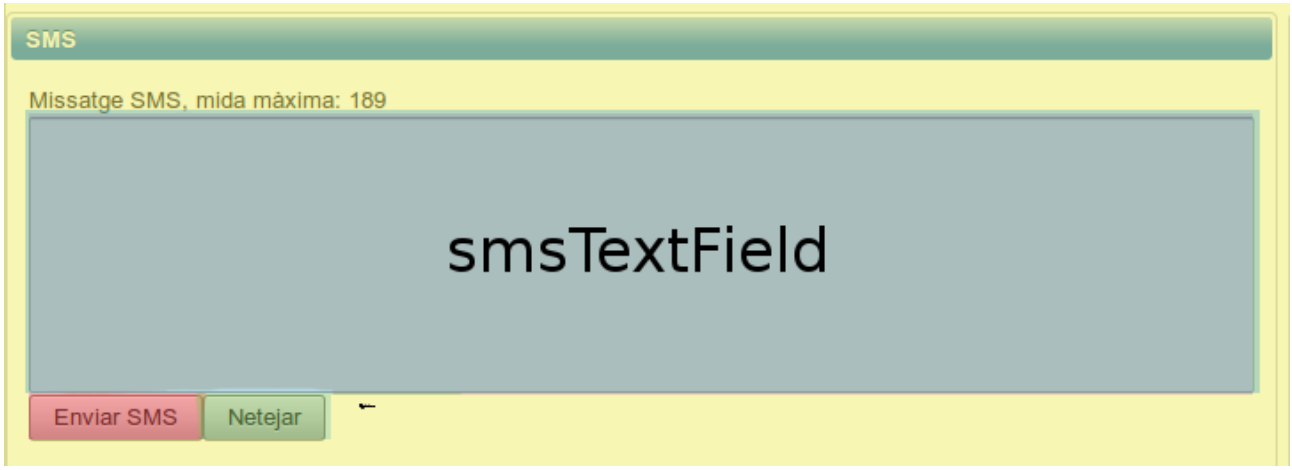


- addStudentButton
  - COMMUNICATIONLAYOUT_ADDSTUDENT_CAPTION (“Afegir estudiant”)

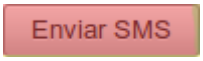


- **smsPanel**

- COMMUNICATIONLAYOUT_SMSTEXTFIELD_CAPTION (“Missatge SMS, mida màxima: ”)



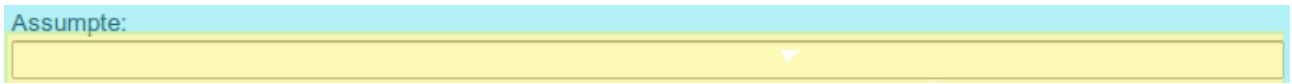
- sendsmsButton
  - COMMUNICATIONLAYOUT_SENDSMS_CAPTION (“Enviar SMS”)



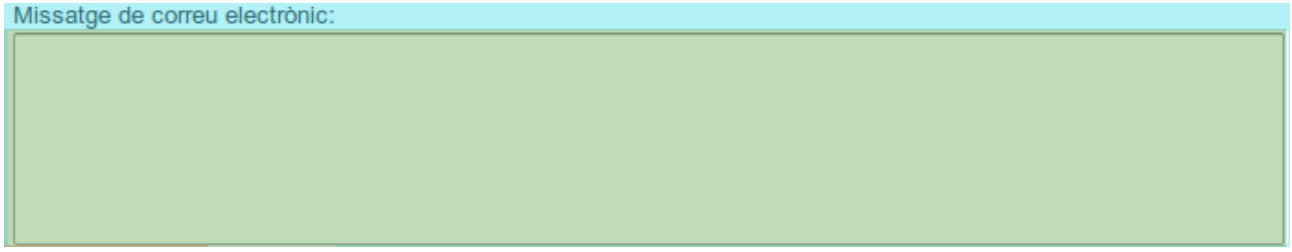
- clearsmsButton
  - COMMUNICATIONLAYOUT_CLEAR_CAPTION (“Netejar”)

- **mailPanel**
  - COMMUNICATIONLAYOUT_MAILPANEL_CAPTION (“Mail”)

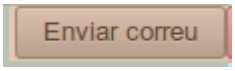
- mailsubjectTextField
  - COMMUNICATIONLAYOUT_MAILSUBJECT_CAPTION (“Assumpte:”)



- mailbodyTextField
  - COMMUNICATIONLAYOUT_MAILBODY_CAPTION (“Missatge de correu electrònic:”)



- sendmailButton
  - COMMUNICATIONLAYOUT_SENDMAIL_CAPTION (“Enviar correu”)



- clearmailButton
  - COMMUNICATIONLAYOUT_CLEAR_CAPTION (“Netejar”)

Aspectes remarcables de la implementació

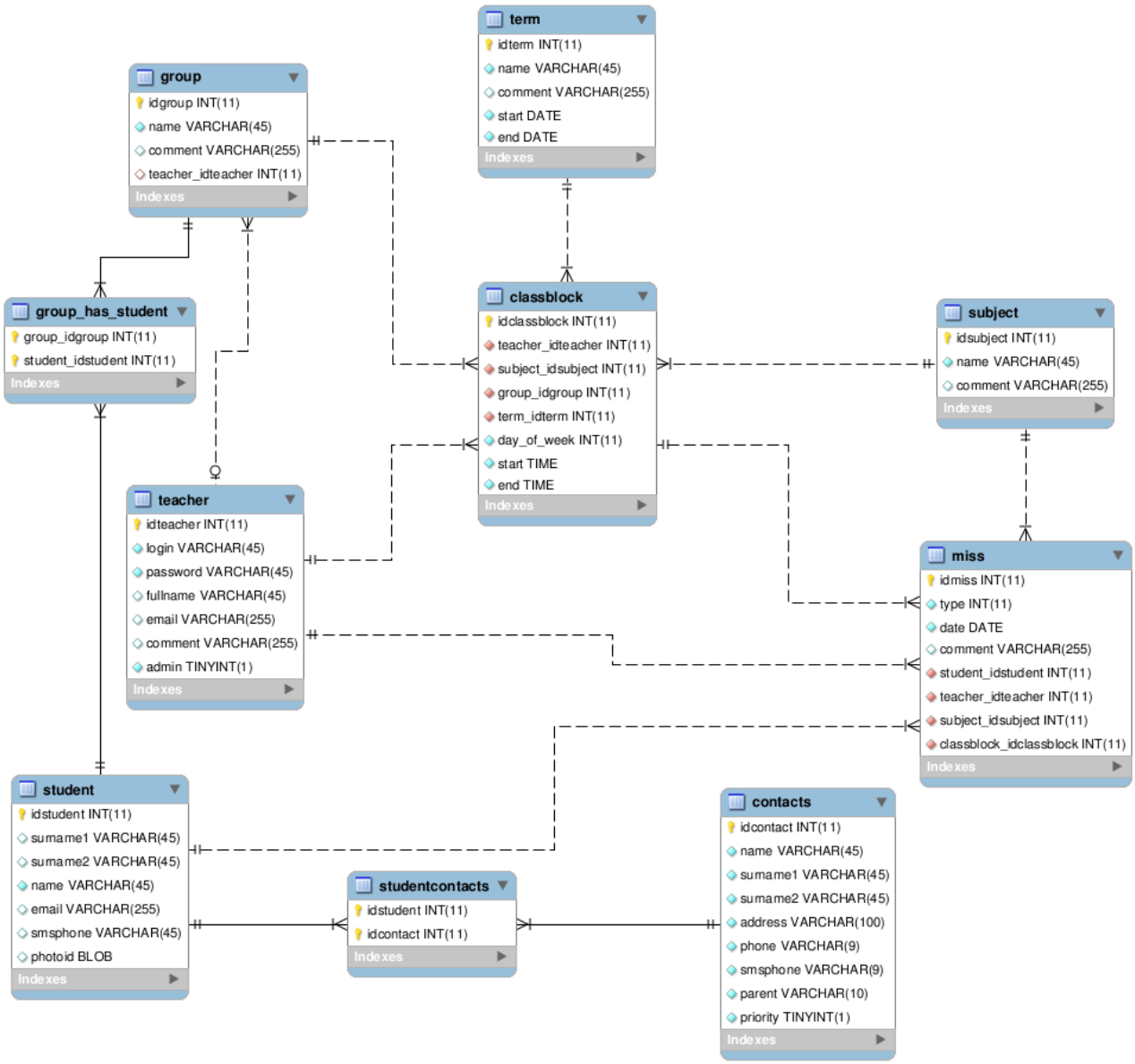
click a l'estudiant de destinationTable

```
                destinationTable.addListener(new Property.ValueChangeListener() {
//-------_>>>>>
                        @SuppressWarnings("rawtypes")
                        @Override
                        public void valueChange(ValueChangeEvent event) {

                                studentContactsTable.removeAllItems();
                                Collection <Student> studentsCollection =
(Collection)destinationTable.getValue() ;
                                boolean multiselect= true;
                                if(studentsCollection.size()>0) {
                                        removeButton.setEnabled(true);
                                        ////////////////////////////////////////////////
                                        //recoge el id y muestra en la tabla los contacts
                                        //pasamos por el bucle de estudiantes
                                        for(Student student: studentsCollection){
                                                Collection <Contact> contactCollection =
app.storage.getContactCollection(student.getID());
                                                //para cada estudiante recojemos su coleccion de contactos
                                                for (Contact contact: contactCollection){
                                                        //si a seleccionado mas de uno, o solo 1
                                                        if (studentsCollection.size()>1){
                                                                multiselect=true;
                                                        }else{
                                                                multiselect=false;
                        }

        refreshContactsTable(studentContactsTable,student,multiselect);
                                                }

                                        }


                                } else {
                                        removeButton.setEnabled(false);
                                }
```

```
                }
        });
```

DataBaseStorage

Model Relacional



Modificacions a la base de dades

1.Creació de dos noves taules: studentcontacts i contact:

- contact: aquesta taula conté la informació dels contactes que poden tenir els estudiants. Consta dels següents camps:
  - idcontact: identificador únic del contacte. Clau primària.
  - name: nom del contacte.
  - surname1: primer cognom del contacte.

- surname2: segon cognom del contacte.
- address: adreça del contacte.
- phone: teléfon del contacte.
- smsphone: teléfon móvil del contacte.
- parent: parentiu del contacte amb l'estudiant.
- priority: nombre que indica si el contacte és prioritari a l'hora de enviar notificacions.

- studentcontacts: taula resultant de la relació entre la taula student i contacts. Per tant consta de les claus primàries de les taules relacionades i d'aquesta manera agrupem els estudiants amb tots els seus contactes. Consta dels següents camps:
  - idstudent: identificador únic de l'estudiant. Clau primària.
  - idcontact: identificador únic del contacte. Clau primària.

2.Modificació de la taula student:

- student: aquesta taula conté la informació dels estudiants. Consta dels següents camps:
  - idstudent: identificador únic de l'estudiant.
  - name: nom de l'estudiant.
  - surname1: primer cognom de l'estudiant.
  - surname2: segon cognom de l'estudiant.
  - email: direcció de correu personal de l'estudiant.
  - smsphone: teléfon mů́il de l'estudiant.
  - photoid: conté l'array de bits de la foto de l'estudiant.

Implementacions a DataBaseStorage

Hem implementat nous métodes tant pel nou disseny de la base de dades com per les modificacions noves a l'aplicació. Els métodes implementats són els següents:

- addStudent: modificació del métode per poder recollir de la sentència SQL el nou camp photoid.

```
public int addStudent(Student student) {

                        int id = 0;
                        if (student.getID()!=0)
                        {
                                Logger.log("DatabaseStorage.addStudent() try to add student with no 0
ID\n" + student);
                                return 0;
                        }
                        try {
                                //Añadido campo photoid para introduccion de blob
                                PreparedStatement st = con.prepareStatement("INSERT INTO
student(surname1,surname2,name,email,smsphone,photoid) "
                                                + " VALUES('"
                                                + student.getSurname1() + "','"
                                                + student.getSurname2() + "','"
                                                + student.getName() + "','"
                                                + student.getEmail() + "','"
                                                + student.getSmsphone()+ "',"
                                                + "?);");
                                //Creacion de fileImputStram recojemos la array de bytes de ese student y
lo intoducimos en la BD
                                ByteArrayInputStream bais=null;
```

```
                                byte[] photoArray = student.getBytePhotoArray();

                                if (photoArray!=null){
                                        bais = new ByteArrayInputStream(photoArray);
                                }
                                st.setBlob(1,bais);
                                st.execute();
                                st.close();
                                ResultSet result = statement.executeQuery("SELECT LAST_INSERT_ID()
AS ID;");
                                result.next();
                                id = result.getInt("ID");
                                student.setID(id);

                        } catch (SQLException e) {
                                e.printStackTrace();
                        }

                        return id;
                }
```

- getStudent: creació del métode poder obtenir un estudiant a partir del seu identificador.

```
public Student getStudent(int idstudent) {
                Student student = null;
                try {
                        ResultSet result = statement.executeQuery("SELECT * FROM student WHERE
idstudent='" + idstudent + "';");
                        while(result.next()) {
                                student = new Student(result.getInt("idstudent"),
result.getString("surname1"),
                                result.getString("surname2"),result.getString("name"),
result.getString("email"),
                                result.getString("smsphone"),result.getBytes("photoid"));
                            }
                } catch (SQLException e) {
                        e.printStackTrace();
                }

                return student;
        }
```

- studentsOfGroup: modificació del métode per poder afegir el camp photoid de l'estudiant. Recull tots els atributs d'un estudiant i crea un objecte que afegeix a una col·lecció que posteriorment és retornada.

```
        public Collection<Student> studentsOfGroup(int idgroup) {
                Vector<Student> vector = new Vector<Student>();
                //añadido photoid
                try {
                        ResultSet result = statement.executeQuery("SELECT idstudent, surname1,
surname2, name, email, smsphone,photoid FROM student, group_has_student
                WHERE group_has_student.group_idgroup = " + idgroup + " AND
group_has_student.student_idstudent = student.idstudent ORDER BY
                surname1,surname2,name;");
                        while(result.next()) {
                                vector.add(new Student(result.getInt("idstudent"),
result.getString("surname1"),
                                result.getString("surname2"), result.getString("name"),
```

```
	result.getString("email"),result.getString("smsphone"),result.getBytes("photoid")));
				}
			} catch (SQLException e) {
				e.printStackTrace();
			}

			return vector;
		}
```

- addContact: creació del métode per poder afegir el contacte a la base de dades i també afegir la identificació de l'estudiant i del contacte a la taula studentContact.

```
public int addContact(StudentContacts contact, int idStudent) {
				int id = 0;
				if (contact.getIdContact()!=0) {
					Logger.log("DatabaseStorage.addStudent() try to add contact with no 0
ID\n" + contact);
					return 0;
				}

				try {

					statement.executeUpdate("INSERT INTO
contacts(name,surname1,surname2,address,phone,smsphone,parent) "
							+ " VALUES('"
							+ contact.getName() + "','"
							+ contact.getSurname1() + "','"
							+ contact.getSurname2() + "','"
							+ contact.getAddress() + "','"
							+ contact.getPhone() + "','"
							+ contact.getSmsphone() + "','"
							+ contact.getParent() + "');");
					ResultSet result = statement.executeQuery("SELECT LAST_INSERT_ID()
AS ID;");
					result.next();
					id = result.getInt("ID");
					contact.setIdContact(id);
				} catch (SQLException e) {
					System.err.println("DatabaseStorage.adContact-> Intro Contacts table
----->>>		"+e.toString());
				}

			try {
					statement.executeUpdate("INSERT INTO
studentcontacts(idstudent,idcontact) "
							+ " VALUES('"
							+ idStudent + "','"
							+ contact.getIdContact() + "');");
					ResultSet result = statement.executeQuery("SELECT LAST_INSERT_ID()
AS ID;");
					result.next();
					id = result.getInt("ID");
					contact.setIdContact(id);
			} catch (SQLException e) {
					System.err.println("DatabaseStorage.adContact-> Intro StudentContacts
table ----->>>	+e.toString());
			}
				return id;
			}
```

- getContactsQueryContainer: retorna un contenidor de consulta dels contactes especificant l'identificador de l'estudiant.

```
public QueryContainer getContactsQueryContainer(int idStudent) {
		QueryContainer studentsQueryContainer = null;
		try {
			studentsQueryContainer = new QueryContainer(
					"SELECT c.idcontact, c.name, c.surname1, c.surname2, c.address,
c.phone, c.smsphone,  c.parent"+
					" FROM contacts c, studentcontacts sc"+
					" WHERE c.idcontact = sc.idcontact and"+
					" sc.idstudent = "+ idStudent +
					" ORDER BY c.surname1, c.surname2, c.name;",con);

		} catch (SQLException e) {
			e.printStackTrace();
		}
		return studentsQueryContainer;
	}
```

- updateContactPriority: creació del métode per poder donar prioritat a un contacte a l'hora d'enviar notificacions. Té com a paràmetre l'estudiant del qual volem enviar la notificació.

```
	public int updateContactPriority(StudentContacts contact){
		int res = 0;
		Integer id= contact.getIdContact();
		String name = contact.getName();
		boolean priority = contact.isPriority();
		String sql = "UPDATE contacts set name ='"+name+"',"
				+ "priority = '"+priority+"'"
				+ "where idcontact="+id+";";
		try{
			res = statement.executeUpdate(sql);
		} catch (SQLException e){
			e.printStackTrace();
		}
		return res;
	}
```

- contactOfstudent: retorna una col·ecció de contactes de l'estudiant especificat mitjançant el seu identificador.

```
	public Collection<StudentContacts> contactsOfStudent(int idStudent) {
		Vector<StudentContacts> vector = new Vector<StudentContacts>();
		//añadido photoid
		try {
			ResultSet result = statement.executeQuery(
					"SELECT c.idcontact, c.name, c.surname1, c.surname2, c.address,
c.phone, c.smsphone, c.parent, c.priority"+
					" FROM contacts c, studentcontacts sc"+
					" WHERE c.idcontact = sc.idcontact and"+
					" sc.idstudent = "+ idStudent +
					" ORDER BY c.surname1, c.surname2, c.name;");
			while(result.next()) {
				vector.add(new StudentContacts(
						result.getInt("idcontact"),
						result.getString("surname1"),
						result.getString("surname2"),
```

```
                                                result.getString("name"),
                                                result.getString("phone"),
                                                result.getString("smsphone"),
                                                result.getString("parent"),
                                                result.getString("priority")));
                        }
                } catch (SQLException e) {
                        e.printStackTrace();
                }
                return vector;
        }
```

- getContact: retorna un contacte a partir del seu identificador.

```
public StudentContacts getContact(int idContact) {
                StudentContacts contact = null;
                try {
                        ResultSet result = statement.executeQuery("SELECT * FROM contacts WHERE
idcontact='" + idContact + "';");
                        while(result.next()) {

                                int resulPriority = result.getInt("priority");
                                Boolean priority=false;

                                if (resulPriority!=0){
                                        priority=true;
                                }
                                contact = new StudentContacts(result.getInt("idcontact"),
                                                result.getString("name"),
                                                result.getString("surname1"),
                                                result.getString("surname2"),
                                                result.getString("address"),
                                                result.getString("phone"),
                                                result.getString("smsphone"),
                                                result.getString("parent"));
                                contact.setPriority(priority);
                        }
                } catch (SQLException e) { e.printStackTrace(); }
                return contact;
        }
```

- isPreviousMiss: retorna un boleà depenent si l'alumne a faltat a clases anteriors.

```
        public boolean isPreviousMiss(int type, Classblock classblock, int studentid, int hours) {
        Boolean exists = false;

        String sql=
                "SELECT *"+
                " FROM miss m, classblock c"+
                " WHERE c.subject_idsubject = m.subject_idsubject"+
                " AND m.classblock_idclassblock not like ("+classblock.getID()+")"+
                " AND student_idstudent =" + studentid +
                " AND m.date = date_format(sysdate(date),'%Y-%m-%d')"+
                " AND date_format(c.end,'%H') = "+classblock.getEnd().getHours() +"-"+hours+
                " AND type = "+ type +";";

        try {
                ResultSet result = statement.executeQuery(sql);
```

```
                if(result.next()) exists = true;

        } catch (SQLException e) {

                System.out.println(e.toString());
        }

        return exists;
        }
```

Possibles Millores

- Millora de la encapsulació de les classes eliminant l'excés de classes incrustades.
- Afegir la possibilitat d'inserir contactes pels estudiants mitjançant fitxers d'extensió csv.
- Afegir un sistema inteligent (estadístic) del càlcul de faltes previes a l'hora seleccionada.
- Edició de la foto de l'estudiant.
- Afegir camp email pels contactes.
- Control d'estandarització del tamany de la foto de l'estudiant (actualment 200×200).
- Possibilitat de que els tutors puguin consultar i editar les faltes de qualsevol de les classes del seu grup.
- Afegir panel de configuració per inserir els nous tipus de falta definits pel professor (possiblitat de definir el color de la falta).
- Afegir llegenda de colors per la visualització dels diferents tipus de falta.