– TFC –

XMNS
APLICACIÓ WEB PER A LA CORRECCIÓ AUTOMÀTICA DE PROVES TIPUS TEST

Memòria

Consultor:Ferran Prados Carrasco

Alumne: Alexis Carrés Ventós

acarres@uoc.edu

Índex

1 Introducció

Primer de tot vull aclarir que XMNS no es tracta d'un acrònim sinó d'una abreviatura utilitzant només les consonants de la paraula exàmens i que pronunciat en anglès (ecs em en es) resulta pràcticament la paraula “exámenes”.

Cada vegada és més habitual trobar-nos llocs web que proposen formació en línia i això els exigeix que ofereixin les eines necessàries com material didàctic, assistència o sistemes d'avaluació o auto-avaluació per a que el servei ofert sigui eficient.

XMNS intenta cobrir una de les necessitats que apareix en la formació en línia. L' avaluació i o auto avaluació de l'alumnat mitjançant proves tipus test.

Aquesta eina, mancant en moltes plataformes de formació en línia, ha de ser útil tant per el docent, per tenir un seguiment i una qualificació de l'alumnat de forma automàtica, com per l'estudiant, que pot conèixer instantàniament el nivell assolit durant el procés d'aprenentatge.
A més, per que l'experiència tant del professorat com dels alumnes sigui més eficient i més pròxima a un entorn presencial, **XMNS** facilita una eina interna de comunicació professor-alumne.

2 Motivació

Actualment ja existeixen eines basades en web i programes d'escriptori que permeten crear proves tipus test auto corregibles i sistemes que permeten una comunicació entre usuaris, sense anar més lluny el mateix correu electrònic. Però després d'una cerca de varies eines ja existents en la web
i amb la experiència adquirida, s'han trobat varis punts que amb aquest treball s'intenta cobrir:

- Facilitat d'ús.

- Possibilitat de realitzar un seguiment general d'una classe.
- Possibilitat de realitzar un seguiment exhaustiu de l'alumnat.
- Comunicació tancada dins de l'aplicació enfocada al procés d'aprenentatge.

Les plataformes analitzades no cobreixen íntegrament els punts anteriorment esmentats ja que, són molt complexes o tot el contrari, són massa simples.

Així és com sorgeix la idea de desenvolupar **XMNS**.

3 Plataformes analitzades

3.1 Eines basades en web

Hot Potatoes

Punts positius:

- Pot generar diferents tipus d`exercicis (tipus test, omplir espais buits, mots encreuats, aparellament o reconstruir frases).
- Auto correcció instantània.
- Pot adjuntar documents per a saber d'on extreure'n informació.

Punts negatius:

- No podem fer seguiment dels alumnes.
- No té una interfície de gestió.

Google Drive (Forms)

Punt positiu:

- Permet introduir fàcilment el qüestionari en una pàgina web.

punt_negatiu:

- Dificultat de veure els resultats obtinguts.

Moodle

Punt positiu:

- Molt complerta i ampliable mitjançant mòduls.

Punts negatius:

- No permet mostrar diferents continguts als alumnes d'una mateixa classe.
- No es pot comunicar dins de la mateixa aplicació amb el professor o sent un professor amb un alumne en concret mitjançant un sistema de correu intern.

3.2 Programes d'escriptori

TestGip

Punts positius:

- Molt complert (excés de funcionalitats).
- Auto correcció instantània.
- Pot adjuntar documents per a saber d'on extreure'n informació.
- Resultats estadístics tant de l'alumne com de la classe en grup.

Punts negatius

- Resultats massa genèrics (S'interpreta més la classe que l'alumne).
- No permet comunicació entre alumne i professor mitjançant l'aplicació.
- Aplicació d'escriptori tant el mòdul alumne com el mòdul professor.

3.3 Webs

Daypo

Punts positius:

- Crear proves tipus test.
- Genera estadístiques dels resultats d'entre tots els alumnes.
- Senzillesa per utilitzar l'eina.

Punts negatius:

- Massa genèric, no es pot fer un seguiment individualitzat.
- Estadístiques molt pobres(temps,encerts).

3.4 Pàgines amb proves d'auto avaluació

w3schools

4 Propòsit

El propòsit de **XMNS** és crear una aplicació web per a la creació, realització i auto-correcció de proves tipus test, a més del seguiment del procés d'aprenentatge a partir de les notes obtingudes pels alumnes i la possibilitat d'enviar i rebre missatges entre els docents i els estudiants.

A més es va trobar interessant utilitzar un Framework MVC en PHP creat per un mateix any enrere modificant-lo i ampliant-lo segons les necessitats del projecte, ja que no es considera aquest treball com a quelcom final sinó com a un prototip. Partint d'aquesta idea és per que s'utilitza aquest Framework i no d'altres ja existents doncs es sacrifica l'eficiència que altres eines poden aportar per la velocitat de desenvolupament del codi.

5 Objectius

Amb aquest treball el que es pretén és generar una plataforma que permeti, en primera instància, poder crear exàmens tipus test per part d'uns professors, que els seus alumnes els responguin i que automàticament es corregeixin.

Els valors afegits de **XMNS** són que, a més, permet una comunicació entre professor i alumnes i una visió del procés d'aprenentatge tant des dels docents com des dels estudiants.

Més concretament, un professor pot:

- Crear exàmens.
- Veure estadístiques de la seva classe en general i de cada alumne en particular.
- Enviar missatges als alumnes de la seva classe.

I un alumne pot:

- Respondre exàmens.
- Veure les estadístiques del seu procés d'aprenentatge.
- Enviar missatges al seu professor.

6 Estudi de viabilitat

Les eines emprades per aquest treball és un ordinador portàtil ACER Aspire 5630 amb sistema operatiu Linux, amb la distribució Ubuntu 12.04 preparat exclusivament per a la realització d'aquest treball i amb el programari necessari per a la seva realització.

Tot el programari utilitzat és de codi lliure.

6.1 Programari

Netbeans 7.3

Es tracta d'un entorn de desenvolupament lliure (IDE), gratuït i sense restriccions d'ús, creat per Sun Microsystems. En un principi va ser creat per a desenvolupar codi per a Java, però en el nostre cas s'ha utilitzat Netbeans for PHP on l'entorn s'enfoca més al desenvolupament en llenguatge PHP gràcies a un potent depurador de codi per a la verificació del codi, l'auto completat i el resultat de codi.

PhpMyAdmin 3.4.10.1

Es tracta d'una eina creada amb PHP, per a l'administració de MySql des d'un navegador.

MysqlWorkbench 5.2.38

És una eina visual per al disseny de base de dades Mysql.

Umbrello UML Modeller 2.8.5

Eina de codi lliure per a crear i editar diagrames UML. Senzill d'utilitzar.

LibreOffice 3.5.7.2

És un conjunt de programes (suite) d'ofimàtica lliure i gratuïta que consta d'un editor de text(Writer), una fulla de càlcul (Calc), un gestor de presentacions (Impress) i un gestor de base de dades (Base).

Firefox 21.0

Navegador web lliure i de codi obert creat per Mozilla. Utilitza el motor d'interpretació Gecko i implementa els actuals i futurs estàndards web.

A més existeix una potent extensió enfocada a la creació de pàgines web anomenada **Firebug.**

6.2 Llenguatges de programació

PHP (PHP Hypertext Pre-processor)

Llenguatge de codi obert del costat del servidor, interpretat per un motor instal·lat normalment sobre un servidor Apache útil per a crear llocs dinàmics i que permet, de forma eficient, la comunicació amb bases de dades Mysql

SQL (*structured query language*)

És un llenguatge declaratiu d'accés a base de dades relacionals

HTML (HyperText Markup Language)

Llenguatge de marques predominant per a l'elaboració de pàgines web, per a traduir la informació i l`estructura en text i per a relacionar-los en forma d'hipertext en un navegador web.

CSS (Cascading Style Sheet)

Llenguatge per a definir la presentació dels elements que formen una pàgina web.

Javascript

És un llenguatge de programació interpretat normalment des del navegador web, basat en objectes i que permet l'accés al DOM (Document object Model) per a modificar-lo.

En aquest punt cal esmentar que no s'utilitza directament javascript en el treball sinó que es farà ús de **jquery,** un framework de javascript que fa més senzilla la seva utilització i de varis mòduls d'aquest framework com **jqplot** per a crear gràfiques o **form** i **validate** per a validar i tractar formularis.

6.3 Sistema Gestor de Base de Dades

MySql 5.5.3

Es tracta d'un sistema de gestió de base de dades (SGBD) multi fil i multi usuari àmpliament utilitzat en aplicacions web.

Està molt lligada a PHP.

6.4 Altres eines emprades

Framework PHP MVC propi

Aquest Framework va ser creat als volts del2010 simplement com a procés d'aprenentatge.

Les característiques principal d'aquest Framework, de forma general, són les següents:

- Segueix el patró MVC (Model-view-controller).
- Per a la connexió a la base de dades utilitza el patró de disseny singleton i una extensió de la ja extensió PDO(PHP Data Objects) per un ús més senzill.
- Dóna més importància a la velocitat en la creació de pàgines que en la seva eficiència.

6.5 Pressupost

No hi ha una despesa extra en programari ja que s'utilitza programari lliure.

L'allotjament de l'aplicació més el cost del domini té un cost conjunt de **50 €** anuals i en el primer any s'adjunta aquesta quantitat en el pressupost final.

Pel treball es dedica, durant un període de 12 setmanes i mitja. Cada setmana es dedicaran 10 hores a la realització del projecte, amb un total de 185 hores.

Aquestes 185 hores es divideixen de la següent manera:

Anàlisis inicial:

Es tracta de decidir el treball a realitzar. Partint d'una idea general imposada, analitzar el mercat actual per a valorar la millora a implementar, fer un estudi de la seva viabilitat i proposar en una primera instància una temporització de les tasques a realitzar en el temps marcat.

L' anàlisis inicial és de gran importància i responsabilitat doncs una mala elecció del treball a realitzar pot condemnar-lo al fracàs abans i tot d'iniciar-lo.

Les modificacions en el concepte inicial que puguin sorgir durant el procés de creació del treball també recauen en aquest perfil.

Aquest procés té una durada total de 40 hores repartides al llarg de 4 setmanes.

Degut a la responsabilitat d'aquesta part, el preu per hora en l'anàlisi inicial és de 35€/h, amb una suma total de **1400€**

Disseny de l'aplicació:

Quan s'accepti el treball és l'hora de l'estudi de la viabilitat del treball i del disseny de l'aplicació.

En aquesta part es decideix el programari i maquinari que s'utilitzen, a més de la generació dels diagrames que posteriorment s'utilitzaran per la codificació.

Les possibles modificacions que puguin sorgir durant el procés de creació del treball són transmeses des d'aquest perfil.

També hi ha certa responsabilitat però en un grau inferior al del procés anterior. Però la experiència és un grau per aquest perfil.

Aquest procés també té una durada de 40 hores repartides al llarg de 4 setmanes.

El preu per hora en el disseny de l'aplicació és de 25€/h, sumant un total de **1000 €**

Implementació de l'aplicació

Fet l'anàlisi i el disseny de l'aplicació és moment d'escriure el codi de l'aplicació.

En aquesta part es dissenyen les interfícies visuals de l'aplicació a més d'escriure el codi. També és el perfil amb el que es fan les proves unitàries per a comprovar el bon funcionament.

Aquest procés és el de menys responsabilitat i té una durada de 5 setmanes, dedicant 10 hores setmanals, seran un total de 50 hores.

El preu per hora en la implementació de l'aplicació és de 35€/h , amb un total de **750 €**

Documentació:

El recull de la documentació durant el procés de realització del treball, la redacció de la memòria final i la preparació de la presentació es realitzen en aquest perfil.

Aquest procés té una durada de 55h repartides durant 5 setmanes i mitja

En aquesta part el preu per hora és de 50€/h, fent un total de **2750€**

Així doncs, el pressupost per a la realització del treball és de **5125 €**

7 Metodologia

La metodologia segueix un model en cascada perquè aquest es pot adaptar perfectament al treball.

Figura 1

En el model en cascada l'inici de cada etapa té que esperar l'acabament de l'etapa anterior; en alguns moments durantaquest treball s'ha hagut de replantejar quelcom de la part del disseny, sobretot en la etapa de implementació.

Els desviaments més destacats apareguts en el transcurs de la realització del treball són els següents:

En quant a la temporització:

- S'ha hagut d'ajustar la temporització que s'havia plantejat en un principi ja que el període de codificació es va haver d'allargar 5 dies.
- Els darrers dies, per motius externs, no es va poder treballar les hores establertes i per tant, per compensar aquestes hores, es va haver d'augmentar les hores setmanals durant les 4 darreres setmanes.

En quant a disseny de l'aplicació:

- En aquest punt comentar que hi han hagut alguns desviaments del plantejament inicial. Els més destacats són:
- En el llistat d'usuaris des de la interfície de l'administrador la ordenació és per rol d'usuari i dins d'aquest, si formen part o no d'una classe (només professors i alumnes).
- Si un professor que forma part d'una classe s'elimina, els exàmens d'aquesta classe no s'eliminen com s'havia plantejat en un principi (quedaran pendents a que s'adjudiqui un nou professor a la classe).
- La única cosa imprescindible al moment de crear una classe és el seu nom, una classe pot no tenir cap professor ni cap alumne en un moment donat.

8 Planificació

Per a poder fer el treball d'una manera eficient s'ha temporitzat la realització de les tasques, a més a més, s'ha plasmat en un diagrama de Gantt per a poder tenir un punt de vista gràfic de les diferents etapes i del temps emprat en cada una d'elles.

1. **Elecció del treball a realitzar.**

   Suposa l'inici oficial marcat pel calendari oficial, tot i que anteriorment ja s'ha reflexionat sobre quin seria el treball a realitzar.

   Inici:27-02-2013
   Fi:27-02-2013

2. **Realització del pla de treball.**

   Inici:01-03-2013
   Fi:09-03-2013

3. **Anàlisis de diferents plataformes formatives.**

   S'analitzen diferents plataformes formatives i aplicacions que proposin proves tipus test, centrant-se sobretot en la seva usabilitat, eficiència i ús que se'n fa dels resultats obtinguts. Es genera un document amb les dades obtingudes.

   Inici:04-03-2013
   Fi:08-03-2013

4. **Escollir la millora a implementar.**

   A partit de l'anàlisi anterior es decideix la millora a implementar que no s'ofereix en les plataformes estudiades.

   Inici:08-03-2013
   Fi:11-03-2013

5. **Realitzar l'especificació i anàlisi de la nova eina.**

   Es genera un document amb l'especificació i anàlisi de la eina a implementar que serà el contingut de la PAC 2. Més concretament s'inclou:

   - Anàlisi dels requeriments no funcionals (rendiment, distribució, seguretat i usabilitat).
   - Anàlisi dels requeriments funcionals (diagrames de classes, model E/R i relacional de la base de dades i casos d'ús).
   - Els actors que es veuen implicats.

   Altre cop es basa amb les dates del calendari oficial.

   Inici:11-03-2013
   Fi:25-03-2013

6. **Realitzar el disseny de l'aplicació.**

   Es genera un document amb el disseny de la eina a implementar que és el contingut de la PAC 3. Més concretament s'inclou.

   - Diagrames d'activitat.
   - Diagrames de seqüència.
   - Generació dels prototips de les principals interfícies.

   Altre cop es basa amb les dates del calendari oficial.

   Inici:25-03-2013
   Fi:12-04-2013

7. **Implementació de l'aplicació.**

   Aquesta part contempla tant la implementació de l'aplicació com les proves de funcionament d'aquesta.

Inici:15-04-2013

Fi:15-05-2013

8. **Realització de la memòria del treball.**

   La memòria és el document on es recullen tots els informes que s'han anat elaborant, més les consideracions finals.

   Inici:10-05-2013

   Fi:24-05-2013

9. **Realització de la presentació del treball.**

   Inici:26-05-2013

   Fi:06-06-2013

8.1 Diagrama de Gantt

En el següent diagrama de Gantt es pot veure com s'han estructurat les diferents etapes d'aquest treball i el temps emprat en cada una d'elles.

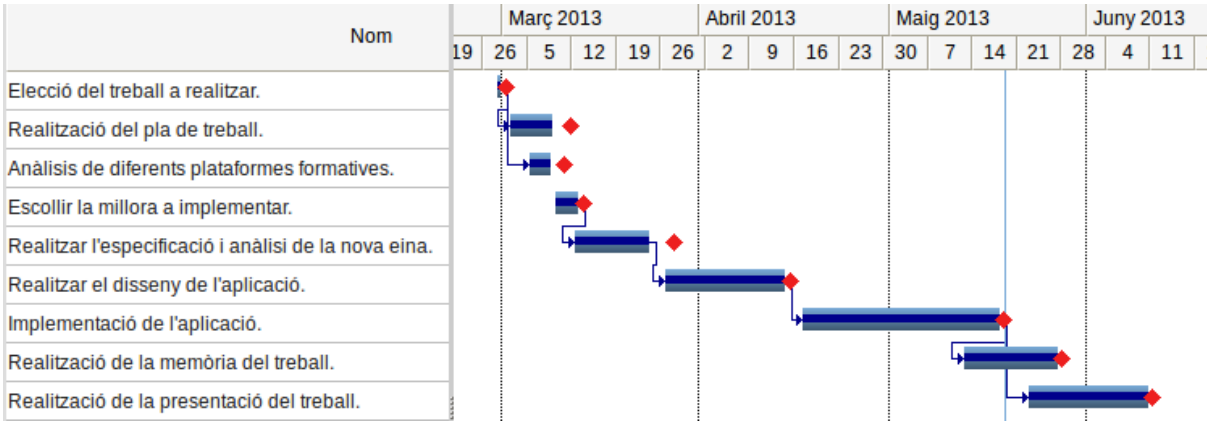

Figura 2

9 Marc de treball i conceptes previs

El marc de treball de **XMNS** s'ha de situar en una aplicació web que en primera instància, pretén poder crear proves tipus test per part d'uns professors per que posteriorment uns alumnes els responguin. Llavors la mateixa aplicació els corregeix.

Amb les notes obtingudes els professors poden tenir una visió del procés d'aprenentatge tant de tota la classe com d'un alumne en particular.

Els alumnes també poden veure el seu progrés de forma individual.

A més, dins de la mateixa aplicació, es permet la comunicació entre professor i alumnes mitjançant missatges de text i la gestió d'aquests missatges.

Tot això amb una interfície molt usable i amb un ràpid procés d'aprenentatge d'ús.

10 Requisits del sistema

10.1 Requisits funcionals

Per a determinar els requisits funcionals del sistema s'ha utilitzat UML (Unified Modeling Language).

UML és un llenguatge gràfic per a visualitzar, especificar, construir i documentar un sistema.

S'usaran els següents diagrames proporcionats per l'estàndard UML per tal d'obtenir el modelatge, anàlisi i disseny de les parts més importants del sistema:

Per a la estructura del sistema:

- Diagrama de classes.

Per al comportament del sistema:

- Diagrames de casos d'ús.
- Diagrames d'activitats.

Per a la interacció:

- Diagrames de seqüència.

També inclou un diagrama Entitat-relació per a modelar les dades i representar les entitats més rellevants del sistema, així com les interaccions entre elles.

10.2 Requisits no funcionals

Rendiment

La correcció de les proves realitzades per l'alumne ha de ser instantània i ràpida, per a aconseguir que l'ús de l'aplicació per a tots els usuaris sigui eficient.

La comunicació entre docents i pupils també ha de ser ràpida i eficient. Es tracta d'una eina interna, és a dir, es descarta l'ús del correu per evitar la pèrdua dels missatges i així, englobar-ho tot dins de la mateixa aplicació.

L'aplicació en conjunt ha de ser neta d'errors ja que és la eina d'avaluació de l'alumne sense la supervisió del professorat en el procés de correcció de les proves.

Distribució

El sistema està pensat per a que estigui allotjat en un servidor Apache amb aquest mínim de característiques:

- PHP5.
- Extensió mod Rewrite.
- Activat short tags.
- En producció oculta els errors
- Possibilitat de comunicar-se amb MYSQL

Com a sistema gestor de base de dades s'utilitza MYSQL, mínim la versió 5 que és l'emprada en el desenvolupament.

L'aplicació està allotjada en un servidor Linux, tant en producció com en desenvolupament. Es pot usar sobre Windows tot i només poder assegurar la bona funcionalitat en servidors Apache.

Es garanteix el bon funcionament de l'aplicació sempre que s'utilitzi Firefox 21.0 sobre Ubuntu, perquè aquesta és la versió del navegador utilitzat en el procés de desenvolupament.

Seguretat

El sistema ha de: filtrar els usuaris registrats mostrant la interfície gràfica corresponent al tipus d'usuari de que es tracta i impedir l'accés a usuaris no registrats en cap de les zones del sistema a excepció de la pàgina d'accés.

També s'impedeix l'accés d'usuaris registrats a zones no permeses pel seu rol.

S'utilitzen mecanismes per encriptar la contrasenya. En concret l'algorisme MD5 per encriptar la clau de l'usuari abans de guardar-la en la base de dades.

Usabilitat

La interfície gràfica per a cada perfil d'usuari ha de ser clara, simple i intuïtiva. No ha de crear confusió i totes les pàgines han de ser accessibles amb la mínima quantitat de clics.

10.3 Actors implicats

Admin

És el rol d'usuari de l'aplicació que pot crear usuaris, adjudicar-els-hi un rol, crear classes i assignar un professor i una llista d'alumnes a una classe.

Professor

És el perfil d'usuari que gestiona una classe, crea les proves, visualitza les estadístiques (tant de la classe que li és assignada com dels alumnes de la classe i de forma individual) i pot enviar missatges als seus alumnes.

Alumne

És el perfil d'usuari que pot respondre als exàmens, visualitzar les estadístiques de les seves proves realitzades i, si és necessari, enviar missatges al professor de la seva classe.

Usuari registrat

És el perfil que pren un usuari quan es registra a l'aplicació i l'administrador encara no li ha adjudicat cap rol. De moment no podrà realitzar cap acció amb la aplicació a excepció d'editar les seves dades (nom d'usurari i contrasenya).

Usuari no registrat

És el perfil que encara no ha estat registrat a l'aplicació i que només pot registrar-se al sistema.

11 Anàlisis i disseny del sistema

11.1 Diagrama entitat-relació

Definits anteriorment els actors que participen en el sistema, anem a mostrar les entitats més rellevants del sistema així com les interaccions entre elles mitjançant un diagrama entitat-relació:

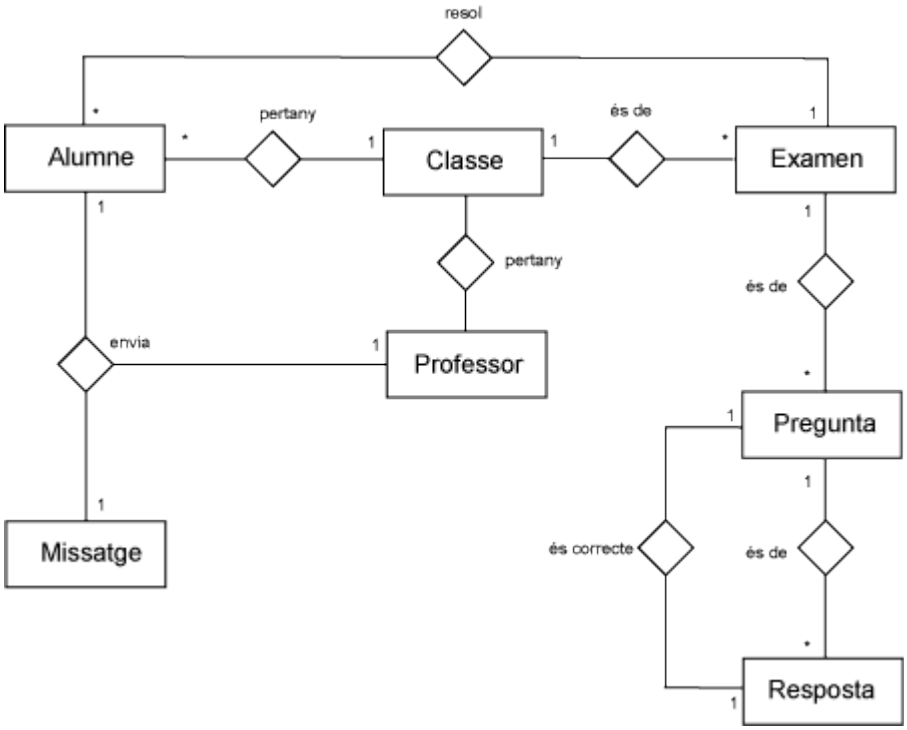

Figura 3

En el diagrama entitat-relació es pot comprovar, que hem limitat que un alumne només pot formar part d'una classe i que un professor només ho pot ser d'una classe.

Creat el diagrama entitat-relació i definit en ell les entitats que formen part del sistema estem preparats per a començar a dissenyar la nostra aplicació.

11.2 Model relacional

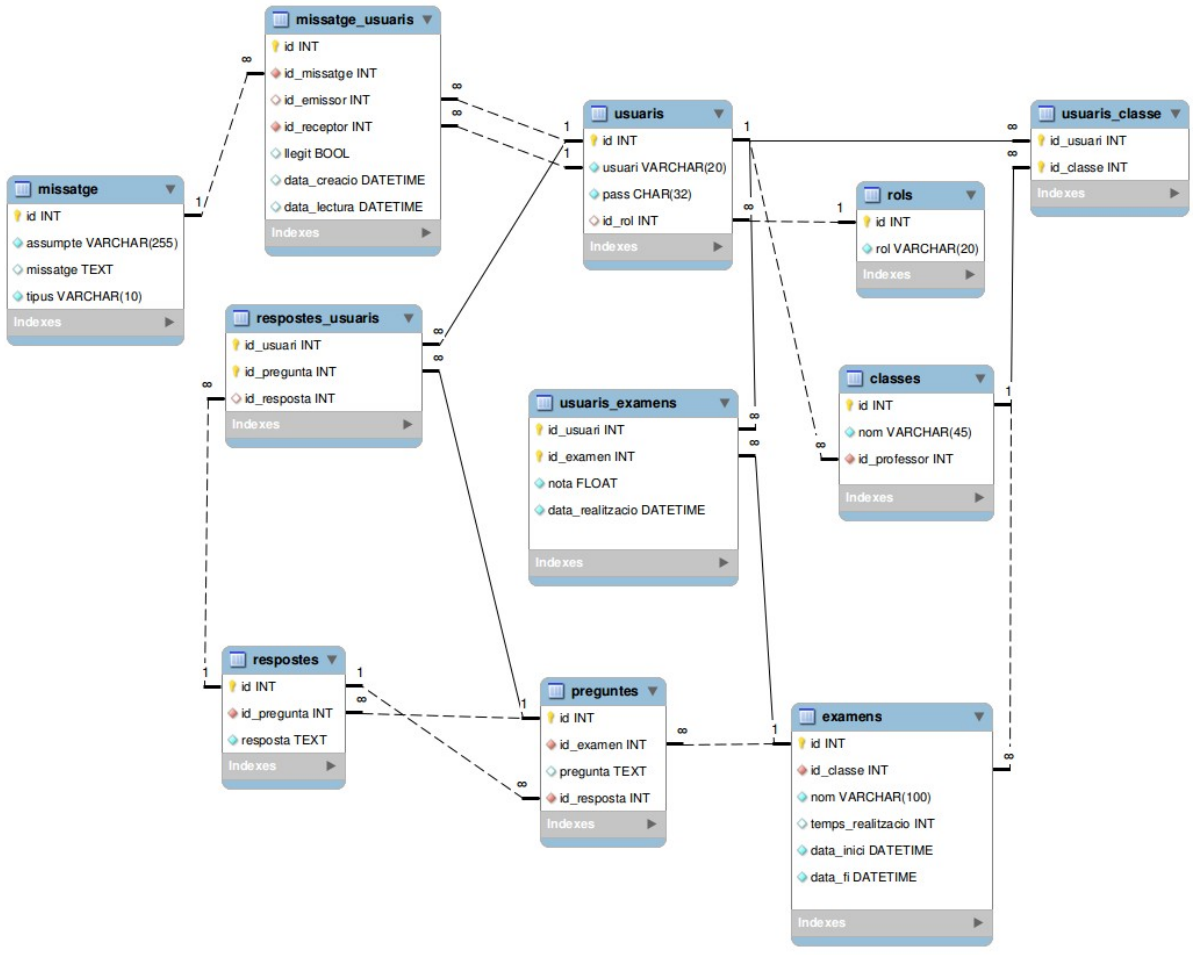

Figura 4

Aquestes són les consultes sql necessàries per a crear la base de dades:

Consultes sql

```
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';

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

-- -----------------------------------------------------
-- Table `examens`.`rols`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `examens`.`rols` (
  `id` INT NOT NULL AUTO_INCREMENT ,
  `rol` VARCHAR(20) NOT NULL ,
  PRIMARY KEY (`id`) ,
  UNIQUE INDEX `rol_UNIQUE` (`rol` ASC) )
ENGINE = InnoDB;

INSERT INTO `examens`.`rols` (rol) VALUES (`admin`),(`alumne`),(`professor`),(`usuari`);
-- -----------------------------------------------------
-- Table `examens`.`usuaris`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `examens`.`usuaris` (
  `id` INT NOT NULL AUTO_INCREMENT ,
  `usuari` VARCHAR(20) NOT NULL ,
  `pass` CHAR(32) NOT NULL ,
  `id_rol` INT NULL ,
  PRIMARY KEY (`id`) ,
  UNIQUE INDEX `nom_UNIQUE` (`usuari` ASC) ,
  INDEX `fk_rol` (`id_rol` ASC) ,
  CONSTRAINT `fk_rol`
    FOREIGN KEY (`id_rol` )
    REFERENCES `examens`.`rols` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

INSERT INTO `examens`.`usuaris` (usuari,pass,id_rol) VALUES (`admin`,`81dc9bdb52d04dc20036dbd8313ed055`,1);
-- -----------------------------------------------------
-- Table `examens`.`classes`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `examens`.`classes` (
  `id` INT NOT NULL AUTO_INCREMENT ,
  `nom` VARCHAR(45) NOT NULL ,
  `id_professor` INT NOT NULL ,
  PRIMARY KEY (`id`) ,
  UNIQUE INDEX `id_professor_UNIQUE` (`id_professor` ASC) ,
  INDEX `fk_professor` (`id_professor` ASC) ,
  CONSTRAINT `fk_professor`
    FOREIGN KEY (`id_professor` )
```

```
    REFERENCES `examens`.`usuaris` (`id` )
    ON DELETE CASCADE
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `examens`.`usuaris_classe`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `examens`.`usuaris_classe` (
  `id_usuari` INT NOT NULL ,
  `id_classe` INT NOT NULL ,
  PRIMARY KEY (`id_usuari`, `id_classe`) ,
  INDEX `fk_usuaris` (`id_usuari` ASC, `id_classe` ASC) ,
  INDEX `fk_classes` (`id_classe` ASC) ,
  CONSTRAINT `fk_usuaris`
    FOREIGN KEY (`id_usuari` , `id_classe` )
    REFERENCES `examens`.`usuaris` (`id` , `id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_classes`
    FOREIGN KEY (`id_classe` )
    REFERENCES `examens`.`classes` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `examens`.`examens`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `examens`.`examens` (
  `id` INT NOT NULL AUTO_INCREMENT ,
  `id_classe` INT NOT NULL ,
  `nom` VARCHAR(100) NOT NULL ,
  `temps_realitzacio` INT NULL ,
  `data_inici` DATETIME NOT NULL ,
  `data_fi` DATETIME NOT NULL ,
  PRIMARY KEY (`id`) ,
  INDEX `fk_examens` (`id_classe` ASC) ,
  CONSTRAINT `fk_examens`
    FOREIGN KEY (`id_classe` )
    REFERENCES `examens`.`classes` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `examens`.`respostes`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `examens`.`respostes` (
  `id` INT NOT NULL AUTO_INCREMENT ,
  `id_pregunta` INT NOT NULL ,
  `resposta` TEXT NOT NULL ,
  PRIMARY KEY (`id`) ,
  INDEX `fk_pregunta` (`id_pregunta` ASC) ,
```

```
  CONSTRAINT `fk_pregunta`
    FOREIGN KEY (`id_pregunta` )
    REFERENCES `examens`.`preguntes` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `examens`.`preguntes`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `examens`.`preguntes` (
  `id` INT NOT NULL AUTO_INCREMENT ,
  `id_examen` INT NOT NULL ,
  `pregunta` TEXT NULL ,
  `id_resposta` INT NOT NULL ,
  PRIMARY KEY (`id`) ,
  INDEX `fk_examen` (`id_examen` ASC) ,
  INDEX `fk_resposta_correcte` (`id_resposta` ASC) ,
  CONSTRAINT `fk_examen`
    FOREIGN KEY (`id_examen` )
    REFERENCES `examens`.`examens` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_resposta_correcte`
    FOREIGN KEY (`id_resposta` )
    REFERENCES `examens`.`respostes` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `examens`.`respostes_usuaris`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `examens`.`respostes_usuaris` (
  `id_usuari` INT NOT NULL ,
  `id_pregunta` INT NOT NULL ,
  `id_resposta` INT NULL ,
  PRIMARY KEY (`id_usuari`, `id_pregunta`) ,
  INDEX `fk_usuari` (`id_usuari` ASC) ,
  INDEX `fk_resposta` (`id_resposta` ASC) ,
  INDEX `fk_pregunta` (`id_pregunta` ASC) ,
  CONSTRAINT `fk_usuari`
    FOREIGN KEY (`id_usuari` )
    REFERENCES `examens`.`usuaris` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_resposta`
    FOREIGN KEY (`id_resposta` )
    REFERENCES `examens`.`respostes` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_pregunta`
    FOREIGN KEY (`id_pregunta` )
    REFERENCES `examens`.`preguntes` (`id` )
    ON DELETE NO ACTION
```

```
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `examens`.`usuaris_examens`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `examens`.`usuaris_examens` (
  `id_usuari` INT NOT NULL ,
  `id_examen` INT NOT NULL ,
  `nota` FLOAT NOT NULL DEFAULT 0 ,
  `data_realitzacio` DATETIME NOT NULL ,
  PRIMARY KEY (`id_usuari`, `id_examen`) ,
  INDEX `fk_usuari` (`id_usuari` ASC) ,
  INDEX `fk_examen` (`id_examen` ASC) ,
  CONSTRAINT `fk_usuari`
    FOREIGN KEY (`id_usuari` )
    REFERENCES `examens`.`usuaris` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_examen`
    FOREIGN KEY (`id_examen` )
    REFERENCES `examens`.`examens` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `examens`.`missatge`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `examens`.`missatge` (
  `id` INT NOT NULL AUTO_INCREMENT ,
  `assumpte` VARCHAR(255) NOT NULL ,
  `missatge` TEXT NULL ,
  `tipus` VARCHAR(10) NOT NULL DEFAULT 'info' ,
  PRIMARY KEY (`id`) )
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `examens`.`missatge_usuaris`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `examens`.`missatge_usuaris` (
  `id` INT NOT NULL AUTO_INCREMENT ,
  `id_missatge` INT NOT NULL ,
  `id_emissor` INT NULL ,
  `id_receptor` INT NOT NULL ,
  `llegit` TINYINT(1) NULL DEFAULT 0 ,
  `data_creacio` DATETIME NULL ,
  `data_lectura` DATETIME NULL ,
  PRIMARY KEY (`id`) ,
  INDEX `fk_emissor` (`id_emissor` ASC) ,
  INDEX `fk_receptor` (`id_receptor` ASC) ,
  INDEX `fk_missatge` (`id_missatge` ASC) ,
  CONSTRAINT `fk_emissor`
    FOREIGN KEY (`id_emissor` )
```

```
    REFERENCES `examens`.`usuaris` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_receptor`
    FOREIGN KEY (`id_receptor` )
    REFERENCES `examens`.`usuaris` (`usuari` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_missatge`
    FOREIGN KEY (`id_missatge` )
    REFERENCES `examens`.`missatge` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;



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

11.3 Casos d'ús

En aquest apartat es dissenyen els casos d'ús més rellevants per tal de cobrir les funcionalitats que ha de tenir l'aplicació.

Per decidir quins són els casos d'ús necessaris, descrivim les funcionalitats que ha de cobrir l'aplicació en llenguatge natural i les dividim en l'actor o actors que hi estan implicats.

“Un usuari no registrat ha de poder registrar-se a l'aplicació i esperar que l'administrador li assigni el rol que tindrà dins de l'aplicació i incloure'l en una classe.

L'administrador crearà les classes necessàries i hi inclourà els alumnes i el professor que en formaran part.

Els professors crearan els exàmens que creguin oportuns per a que, durant el període que hagi decidit, els alumnes el resolguin.

Si un alumne no resol l'examen, aquest es considerarà fet però suspès amb la nota mínima.

En qualsevol moment un alumne pot enviar un missatge al professor i un professor pot enviar un missatge als alumnes que ell esculli.

En qualsevol moment un professor podrà consultar les estadístiques de la seva classe en general o d'un alumne en particular i un alumne podrà veure les seves pròpies estadístiques.”

Analitzant el text anterior podem extreure'n que els requeriments de l'aplicació són els següents:

- Ha de permetre registrar-s'hi.
- Per part de l'administrador:
    - Ha de permetre gestionar els diferents tipus d'usuaris que faran ús de l'aplicació.
    - Ha de permetre crear i gestionar classes.
- Per part del professor:
    - Ha de permetre crear exàmens.
    - Ha de permetre veure les estadístiques generades per els usuaris tant a nivell de classe com a nivell d'alumne de la classe.
- Per part de l'alumne:
    - Ha de permetre respondre les preguntes dels exàmens.
    - Ha de permetre veure les estadístiques generades pel fet d'haver respost els exàmens.

A més a més ha de permetre enviar missatges tant des d'un professor a l'alumne o alumnes de la seva classe com des d'un alumne al seu professor.

Per a poder complir amb els requeriments, a continuació enumero les accions que el sistema pot realitzar juntament amb els actors que hi estan implicats.

- **Registrar usuari a l'aplicació** (usuari no registrat)
- **Accedir a l'aplicació** (usuari registrat, admin, professor i alumne)
- **Sortir de l'aplicació** (usuari registrat, admin, professor i alumne)
- **LListar usuaris** (admin)
- **Canviar de rol a un usuari registrat** (admin)
- **Eliminar usuari** (admin)
- **Crear classe** (admin)
- **Eliminar classe** (admin)
- **Assignar professor a una classe** (admin)
- **Canviar el professor d'una classe** (admin)
- **Assignar alumnes a una classe** (admin)

- **Crear exàmens** (professor)
- **Crear preguntes** (professor)
- **Publicar exàmens** (professors)
- **Llistar exàmens** (professor,alumne)
- **Editar dades de usuari** (usuari registrat, admin, professor, alumne)
- **Enviar missatge** (professor,alumne)
- **Marcar missatge com a llegit** (admin,professor,alumne)
- **Respondre exàmens** (alumne)

Diagrama i fitxes de classes d'us

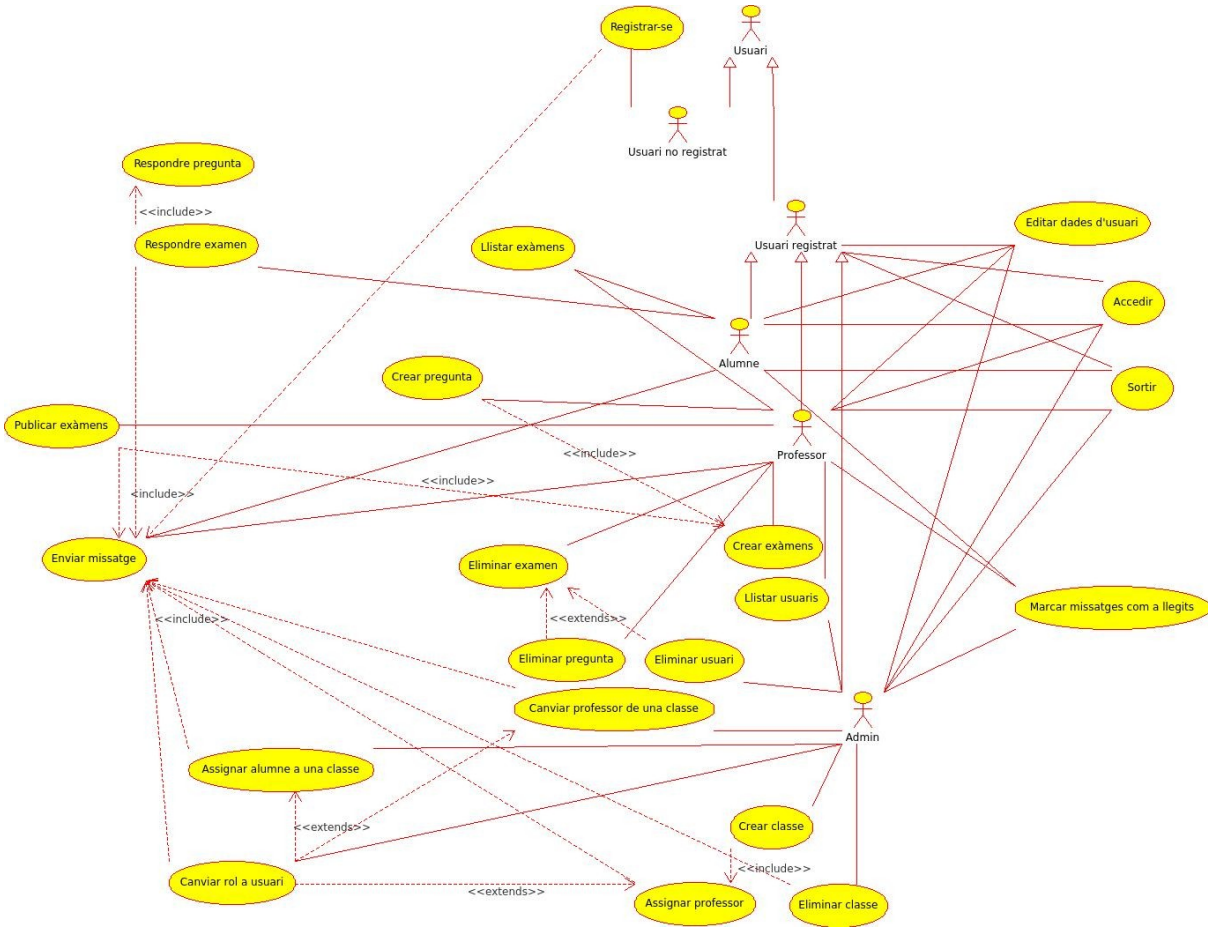

Figura 5

Nom	Registrar-se
Autor	Alexis Carrés Ventós
Actors	Usuari no registrat, Usuari registrat
Casos d'ús inclosos	Enviar missatge
Casos d'ús relacionats	
Descripció	Un futur usuari es dona d'alta a l'aplicació, es comprova que no existeixi el nom de usuari entrat i si és així queda l'usuari registrat però sense cap rol assignat
Pre condicions	No és encara un usuari registrat.
Flux normal	1-L'usuari no registrat entra les seves dades (nom,contrasenya) 2-L'usuari no registrat passa a ser usuari registrat. 3-Es mostra l' entorn d'un usuari no registrat on es dona un missatge de benvinguda.
Flux alternatiu	2-A Les dades subministrades no són vàlides, és a dir: • No ha entrat un nom d'usuari. • El nom de l'usuari entrat té menys de 3 caràcters • El nom de l'usuari entrat ja existeix com a nom d'usuari • No ha entrat una contrasenya • La contrasenya té menys de 6 caràcters. 3-A Torna al punt 1 del flux normal
Post condició	L'usuari no registrat ara és un usuari registrat però sense cap rol assignat.

Nom	Accedir
Autor	Alexis Carrés Ventós
Actors	Usuari registrat, Admin, Professor, Alumne
Casos d'ús inclosos	
Casos d'ús relacionats	
Descripció	L'usuari entra les seves dades per a accedir al sistema
Pre condicions	L'actor es tracta d'un usuari registrat, Admin, Professor o Alumne
Flux normal	1-L'actor entra les seves dades (nom,contrasenya) 2-L'usuari no registrat passa a ser usuari registrat. 3-Es mostra l' entorn corresponent al tipus d'actor del que es tracta
Flux alternatiu	2-A Les dades subministrades no són vàlides, és a dir: • No ha entrat un nom d'usuari. • No ha entrat una contrasenya • La contrasenya té menys de 6 caràcters. 3-A Torna al punt 1 del flux normal 2-B No existeix cap usuari amb les dades subministrades. 3-B S'invita amb un missatge a registrar-se a l'aplicació
Post condició	L'usuari queda amb sessió iniciada al sistema.

Nom	Sortir
Autor	Alexis Carrés Ventós
Actors	Usuari registrat, Admin, Professor, Alumne
Casos d'ús inclosos	
Casos d'ús relacionats	
Descripció	L'usuari surt del sistema
Pre condicions	L'actor es tracta d'un usuari registrat, Admin, Professor o Alumne i anteriorment ha accedit a l'aplicació
Flux normal	1-L'actor selecciona sortir de l'aplicació
Flux alternatiu	
Post condició	L'usuari surt del sistema tancant la sessió i queda a la pàgina inicial d'accés o registre a l'aplicació.

Nom	Llistar usuaris
Autor	Alexis Carrés Ventós
Actors	Admin, Professor
Casos d'ús inclosos	
Casos d'ús relacionats	
Descripció	Consultar usuaris del sistema depenent si es tracta d'un actor Admin o d'un actor Professor aquesta llista serà diferent. <u>Admin</u> Mostra: • la llista de tots els usuaris amb la classe de la que formen part, dividits pel rol que tenen, per si són professors o alumnes i per si formen o no part d'una classe. Les accions que es permetran fer des d'aquest punt són: usuari registrat sense rol ni classe assignada: • Canviar rol a usuari • eliminar usuari altres: • Veure característiques de l'usuari • eliminar usuari <u>Professor</u> Mostra: • els alumnes que se'ls ha assignat la seva classe. Les accions que es permetran fer des d'aquest punt són: • Veure estadístiques d'alumne
Pre condicions	Es tracta d'un actor Admin o Professor i hi ha resultats a mostrar
Flux normal	1-Mostra la llista d'usuaris que té accés depenent del tipus d'actor del que es tracta.
Flux alternatiu	1-No hi ha resultats a mostrar i mostra el missatge corresponent
Post condició	Tenim la llista d'usuaris corresponent, diferent depenent del tipus d'actor.

Nom	Canviar rol a usuari
Autor	Alexis Carrés Ventós
Actors	Admin
Casos d'ús inclosos	Enviar missatge
Casos d'ús relacionats	
Descripció	Si es tracta d'un usuari ja registrat podem canviar-li el seu rol com a usuari de l'aplicació.
Pre condicions	L'usuari a qui es pretén canviar de rol no forma part de cap classe (Professor ni Alumne) que ja s'hagi iniciat, és a dir que encara no s'hagi publicat cap examen.
Flux normal	1-Seleccionem el nou rol que ha de prendre l'usuari a qui li volem adjudicar un nou rol. 2-Enviem un missatge a l'usuari anunciant-li el rol que li ha estat adjudicat
Flux alternatiu	
Post condició	L'usuari escollit té el nou rol i ha rebut el missatge de confirmació.

Nom	Eliminar usuari
Autor	Alexis Carrés Ventós
Actors	Admin
Casos d'ús inclosos	
Casos d'ús relacionats	
Descripció	Opció que permet eliminar a un usuari del sistema i tot el que implica aquest fet, és a dir: • Si es tracta d'un professor sense classe assignada, d'un alumne sense classe assignada o d'un usuari registrat, se li elimina les seves dades i els seus possibles missatges tant enviats com rebuts. • Si es tracta d'un professor amb una classe assignada s'eliminen els missatges tant enviats com rebuts, i la classe queda sense professor però activa. • Si es tracta d'un alumne que ja té una classe assignada s'eliminen els missatges tant enviats com rebuts i els exàmens realitzats. Si l'usuari tenia sessió iniciada, en la següent acció que realitzi és enviat a la pàgina d'accés i registre.
Pre condicions	L'usuari a eliminar no es tracta de l'Admin
Flux normal	1-Eliminem l'usuari a eliminar. Usuari registrat: 2-Esborrem els missatges que ha pogut rebre. Alumne sense classe assignada: 2-Esborrem els missatges que ha pogut rebre. Professor sense classe assignada: 2-Esborrem els missatges que ha pogut rebre. Alumne amb classe assignada: 2-Esborrem els missatges que ha pogut rebre i enviar.

	3-Eliminem les possibles respostes que hagi pogut contestar. Professor amb classe assignada: 2-Esborrem els missatges que ha pogut rebre i enviar.
Flux alternatiu	
Post condició	No queda cap rastre en el sistema de l'antic usuari.

Nom	Crear classe
Autor	Alexis Carrés Ventós
Actors	Admin
Casos d'ús inclosos	Assignar Professor,Canviar professor d'una classe. Assignar alumne una classe
Casos d'ús relacionats	
Descripció	Es crea una nova classe
Pre condicions	La classe no existeix
Flux normal	1-Introduïm un nom de classe 2-Seleccionem, si volem, el professor disponible entre els professors ja assignats o els usuaris que encara no tenen cap rol assignat. 3-Seleccionem, si volem, els alumnes que formaran part de la classe excloent el professor si s'ha seleccionat (es tractava d'un usuari sense rol assignat) i permetent als alumnes sense classe assignada i als usuaris sense rol assignat.
Flux alternatiu	El nom de la classe ja existeix 2-A Anunciem que el nom ja existeix i permetem que assigni un altre nom i retornem al pas 1
Post condició	La classe és creada

Nom	Assignar professor
Autor	Alexis Carrés Ventós
Actors	Admin
Casos d'ús inclosos	Enviar missatge
Casos d'ús relacionats	Canviar rol a usuari
Descripció	S'assigna un professor a una classe que s'està a punt de crear
Pre condicions	L'actor que serà professor o és un usuari registrat o bé és un professor que no té encara cap classe assignada
Flux normal	1-Seleccionem un usuari de una llista, que no sigui alumne, per a que sigui el professor de la classe. 2-Enviem un missatge al nou professor de la classe
Flux alternatiu	
Post condició	Un professor ha estat assignat a una classe

Nom	Canviar professor de una classe
Autor	Alexis Carrés Ventós
Actors	Admin
Casos d'ús inclosos	Enviar missatge
Casos d'ús relacionats	Canviar rol a usuari
Descripció	Permetre canviar el professor assignat d'una classe per un altre.
Pre condicions	El nou professor de la classe ha de ser: • Un usuari sense cap rol assignat. • Un professor sense cap classe assignada.
Flux normal	1-Seleccionem un usuari d'una llista 2-Enviem un missatge al professor anunciant-li la nova classe que té assignada 3-Enviem un missatge als alumnes que estan en aquesta classe
Flux alternatiu	
Post condició	A la classe se li ha canviat el professor que es tenia assignat

Nom	Assignar un alumne a una classe
Autor	Alexis Carrés Ventós
Actors	Admin
Casos d'ús inclosos	Enviar missatge
Casos d'ús relacionats	Canviar rol a usuari
Descripció	Permetre assignar un alumne a una classe o bé canviar de classe si aquestes encara no han publicat exàmens. També es pot assignar directament un usuari sense rol a una classe.
Pre condicions	La classe a on es vol assignar l'alumne encara no ha d'haver publicat exàmens. L'usuari, si pertany a una classe, ja sigui alumne o professor, les classes encara no han d'haver publicat cap examen.
Flux normal	1-Seleccionem la nova classe que se li assignarà al nou alumne. 2-Enviem un missatge a l'alumne que s'ha canviat de classe o que s'ha assignat a una classe.
Flux alternatiu	
Post condició	Tenim un nou alumne a la classe seleccionada

Nom	Crear examen
Autor	Alexis Carrés Ventós
Actors	Professor
Casos d'ús inclosos	Crear pregunta,Publicar examen
Casos d'ús relacionats	
Descripció	Un professor crea un examen
Pre condicions	
Flux normal	1-Introduïm un nom per al nou examen.

	2-Introduïm la data d'inici per a l'examen 3-Introduïm la data màxima de realització de l'examen per part dels alumnes. 4-Introduïm el temps que es necessitarà màxim per a respondre l'examen (30, 60 o 120 segons) 5-Creem preguntes 6-Publiquem l'examen
Flux alternatiu	2-A Si el nom d'aquest examen ja existeix dins de la classe, s'avisa per que esculli un altre nom. 3-B Si la data d'inici és igual o superior a la data final s'avisa al professor per que canvi les dates 6- C Si no s'han creat preguntes es convida al professor a crear-ne
Post condició	Tenim un nou examen preparat per relacionar-li preguntes

Nom	Crear pregunta
Autor	Alexis Carrés Ventós
Actors	Professor
Casos d'ús inclosos	
Casos d'ús relacionats	
Descripció	Un professor crear una pregunta per a un examen
Pre condicions	
Flux normal	1-Introduïm el text de la pregunta 2-Es selecciona crear resposta per introduir el text de la resposta 3-Mentre volem respostes realitzem el pas 2 4-Marquem la resposta correcte,
Flux alternatiu	2-A Si no s'ha introduït un text de pregunta s'avisa al professor 3-B Si no s'ha introduït un text de resposta s'avisa al professor 6-C Si no s'ha marcat una resposta com a correcte s'avisa al professor
Post condició	Tenim un nova pregunta preparada per relacionar-li preguntes

Nom	Publicar examen
Autor	Alexis Carrés Ventós
Actors	Professor
Casos d'ús inclosos	Enviar missatge
Casos d'ús relacionats	
Descripció	Un professor publica un examen per a que els alumnes puguin respondre'l entre les dates
Pre condicions	S'ha creat un examen i com a mínim té una pregunta vinculada
Flux normal	1-Enviem un missatge als alumnes que estan associats a aquesta classe
Flux alternatiu	
Post condició	Tenim un nou examen publicat preparat per a que els alumnes de la classe responguin les seves preguntes.

Nom	Editar dades d'usuari
Autor	Alexis Carrés Ventós
Actors	Usuari registrar, Admin, Professor,Alumne
Casos d'ús inclosos	
Casos d'ús relacionats	
Descripció	Possibilitat de canviar les seves dades de usuari (nom i contrasenya)
Pre condicions	
Flux normal	1-L'Actor implicat decideix canviar alguna de les seves dades.
Flux alternatiu	2-A Les dades subministrades no són vàlides, és a dir: • No ha entrat un nom de usuari. • El nom de l'usuari entrat ja existeix com a nom d'usuari • El nom nou té menys de 3 caràcters • No ha entrat una contrasenya • La contrasenya té menys de 6 caràcters. 3-A Torna al punt 1 del flux normal
Post condició	Les dades de l'usuari han estat editades.

Nom	Enviar missatge
Autor	Alexis Carrés Ventós
Actors	Professor,Alumne
Casos d'ús inclosos	
Casos d'ús relacionats	
Descripció	Possibilitat d'enviar missatges entre els actors implicats. Els professors enviaran missatges a alumnes i els alumnes podran enviar-ne al seu professor. Hi ha certes accions que també poden enviar missatges.
Pre condicions	
Flux normal	1-L'actor escriu l'assumpte del missatge. 2- L'actor escriu el text del missatge. Si l'actor és un professor Professor: 3- El professor selecciona els alumnes a qui se'ls ha de fer arribar el missatge 4-S'envia el missatge.
Flux alternatiu	4-A Si no s'ha escrit un assumpte s'avisa a l'actor 4-B Si no s'ha escrit cap missatge s'avisa a l'actor 4-C Si no s'ha escollit cap receptor s'avisa a lactor
Post condició	El missatge ha estat enviat.

Nom	Marcar missatge com a llegit
Autor	Alexis Carrés Ventós

Actors	Professor,Alumne
Casos d'ús inclosos	
Casos d'ús relacionats	
Descripció	Quan un actor involucrat decideix que ja ha llegit el missatge, pot marcar el missatge com a llegit, Aquesta acció realment no esborra el missatge, només queda marcat com a llegit i l'actor no el visualitza més.
Pre condicions	Hi ha algun missatge per ser llegit
Flux normal	1-L'actor llegeix el missatge i el marca com a llegit 2-El missatge es guarda amb la data amb que s'ha llegit
Flux alternatiu	
Post condició	El missatge ja es visualitza a la llista de missatges com a llegit.

Nom	Respondre examen
Autor	Alexis Carrés Ventós
Actors	Alumne
Casos d'ús inclosos	
Casos d'ús relacionats	
Descripció	Un usuari selecciona respondre un examen. Quan això succeeix el comptador comença a descomptar i l'alumne té que respondre les preguntes de l'examen
Pre condicions	
Flux normal	1-L'alumne selecciona les respostes de l'examen mentre no s'ha esgotat el temps. 2-L'alumne decideix finalitzar l'examen 3-S'auto corregeix l'examen i se li dona una nota.
Flux alternatiu	4-B Si s'ha esgotat el temps, s'acaba la possibilitat de respondre l'examen i i les preguntes no contestades queden com a errònies
Post condició	

12 Implementació i proves

12.1 Característiques del Framework php MVC propi

En aquest apartat s'explica de manera ràpida i amb exemples de codi, el funcionament del Framework.

L'estructura dels directoris és la següent:

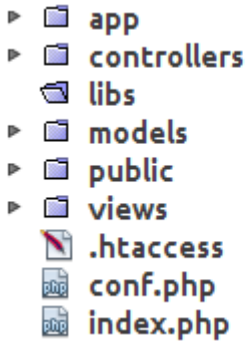

Figura 6

app:És on hi ha els fitxers propis i intocables per al funcionament del Framework (a excepció de .htaccess i index.php)

controllers:Directori on s'ubiquen els controladors.

models:Directori on s'ubiquen els models.

views:Directori on s'ubiquen les vistes. També s'hi trobea la plantilla bàsica on s'inclouran les vistes i els fitxers css i javascript exclusius si és de cada controlador i les possibles imatges.

public: Els fitxers css i javascript i les imatges comunes en tota l'aplicació.

conf.php:És el fitxer de configuració on s'indicaran els mètodes dels controladors que es cridaran per defecte si no es passa cap en la url, el domini on s'allotjarà l'aplicació i els paràmetres de connexió a la base de dades.

Per a poder utilitzar el patró MVC requerim un únic punt d'entrada a l'aplicació, aquest és el fitxer **index.php** que hi ha a l'arrel.

Per a poder realitzar aquesta acció utilitzem un fitxer .**htaccess** que els servidors Apache proporcionen, i gràcies a l'extensió mod Rewrite, redirigim al fitxer index.php anteriorment esmentat.

Fitxer .htaccess

```
RewriteEngine on

RewriteCond %{REQUEST_FILENAME} !\.(js|ico|gif|jpg|jpeg|png|css|swf|xml|txt)$
RewriteCond %{REQUEST_FILENAME} !-d
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-l

RewriteRule ^(.*).html$ index.php?url=$1 [QSA,L]
```

Com podem comprovar s'envien totes les peticions amb extensió .**html** que es facin, i que no siguin fitxers amb certes extensions ni directoris, cap al fitxer **index,php** amb paràmetres **get.**

En el fitxer **index.php** d'entrada realitzem varies coses:

- Iniciem varies constants que ens seran útils en la nostra aplicació.
- Incloem a les rutes on cercar alguns dels fitxers que ens seran necessaris en l'aplicació.
- Incloem el fitxer **autoload.php.**
- Executem la classe MVC.

Autoload

El fitxer **autoload.php** ens permet no haver d'incloure les classes que en un moment determinat necessitem mitjançant la funció màgica __autoload()

Autoload.php

```
function __autoload($clase) {
   include_once $clase . '.php';
}
```

Classe MVC

Aquesta classe és la que realitza tota la màgia del patró mvc dividint la url rebuda en la petició des del navegador cap al controlador demanat, el mètode del controlador i si requereix, els paràmetres que se li passen al mètode.

Per exemple, la url **www.alec6.com/xmns/admin/editar-usuari/3.html** ens duria al controlador AdminController i executaria el mètode editarUsuari passant-li com a paràmetre el valor 3

Es veuen certs convenis establerts i que s'han de complir sempre:

Els controladors s'escriuen la primera en majúscules i acabats amb el sufix Controller. Això passarà igual amb els models, però en el seu lloc el sufix serà Model.

En el cas de les vistes això no succeirà, la característica serà que aquests tindran com a extensió .**phtml**

Seguretat

Com anteriorment hem comentat, quan un usuari es registra la seva contrasenya queda encriptada mitjançant MD5 a la base de dades. Posteriorment, quan es vol accedir a l'aplicació es comprova que l'usuari estigui registrat, i si és així, es comprova quin rol té i si té els permisos adients per a accedir a la pàgina que s'està intentant entrar.

Anem a explicar aquest procés:

Suposant que ja es tracta d'un usuari registrat, l'aplicació ho comprova i guarda en sessió les dades necessàries referents al usuari (id, nom i rol) i l'enviem a la pàgina d'entrada de l'usuari registrat.

Comprovació d' usuari registrat, registre de la sessió d'usuari i missatge de usuari correcte amb el seu id de rol

```
(...)
            case 'entrar':
               if($model_usuaris->existsUser($user,$pass)){
                  $usuari = $model_usuaris->getUser($user,$pass);
                  if($usuari){
                      $model_rol = $this->model('rols');
                     $this->session->set('user',$usuari->usuari);
                     $this->session->set('rol',$usuari->id_rol);
                     $this->session->set('id',$model_usuaris->getIdUserByUser($user));
                      //És correcte!!, un cop registrat en sessió l'usuari i el rol reenviem a la
pàgina inicial del usuari
                     echo  "1#".BASE_URL. $model_rol->getRolById($usuari->id_rol)->rol;
                     exit();
(...)

//Mètode dins del model UsuarisModel
   /**
    * Mètode per comprovar si un nom d'usuari ja existeix a la base de dades
    * @param string $name
    * @param string $pass
    * @return boolean
    */
   public function existsUser($name,$pass=NULL)
   {
      $where_pass='';
      if($pass){
         $this->prepararVariable('pass',md5($pass));
         $where_pass = " AND pass=:pass ";
      }
      $this->prepararVariable('usuari',$name);
      $sql = "SELECT usuari FROM usuaris WHERE usuari=:usuari $where_pass LIMIT 1";
```

```
        $resposta = $this->novaConsulta($sql);
        return (boolean) $resposta;
    }
```

Cada intent d'accés a una pàgina de l'aplicació (a excepció de la pàgina inicial) es comprova que té registrada la sessió i posteriorment si té permís per a veure la pàgina sol·licitada, i en cas negatiu s'envia a la pàgina inicial.

Decisió de si té permès l'accés a la pàgina sol·licitada

```
//Mètode constuctor del controlador AlumneController que exten de Indexcontroller
public function __construct(){
          parent::__construct();
          $model_usuari = $this->model('usuaris');
          $model_usuaris_classe=$this->model('usuarisClasse');

          if($model_usuari->getUserById($this->session->get('id'))->id_rol !='3'){
             Url::redir();
          }

                    $classe = $model_usuaris_classe->getClasseByIdUser($this->session-
>get('id'));

          $this->variables['id_classe']=$classe?$classe[0]->id_classe:$classe;
          $this->variables['usuari']=$this->session->get('user');
          $this->variables['id_usuari']=$this->session->get('id');

          $this->view->addjs('alumne','alumne');
    }

// constructor de la classe IndexController
    public function __construct() {
         $this->session = new Session;
        parent::__construct();
    }

//Part del mètode index del controlador IndexController
(...)
        //Si ja està loguejat l'enviem a la pàgina principal del seu entorn
        if($this->session->get('user') && $this->session->get('id')){
           $model_usuari = $this->model('usuaris');
                $this->redireccionar($model_usuari->getUserById($this->session->get('id'))-
>id_rol);
        }
        //Altrament mostrem la pàgina de login
        $this->variables['title']="Pàgina d'accés";

        $this->min('login',0);
        $this->view->showView('login',$this->variables);
(...)
```

Ajudes

El motiu de l'ús d'aquest Framework és la velocitat a l'hora de programar per sobre de l'eficiència de l'aplicació. A continuació s'expliquen amb exemples algunes de les característiques pròpies del framework utilitzat:

Helpers: Són un conjunt de classes amb mètodes estàtics que es localitzen dins del directori **app/base/helpers**

N'existeixen 6:

- **Variables**: Per fer un filtrat de les variables **get i post** rebudes.
- **Url**: Per fer diferents tractaments de les urls.
- **Strings**: Per fer tractaments especials a cadenes de text com per exemple retallar strings.
- **Date**: Per fer tractaments especials amb dates.
- **Session**: Per encapsular els mètodes per a treballar amb sessions.
- **Debug**: Per mostrar resultats de variables en un moment determinat.

Juntament amb el depurador de codi incorporat a Netbeans (IDE utilitzada) disposem d'una gran eina de depuració de codi.

Aquest Helper s'utilitza molt durant el procés de codificació, per tant es mereix mostrar-lo tot.

Classe debug

```
<?php
class Debug
{
   /**
    * Mètodes per a debugar
    */
  /**
   * Ens guarda la resposta en un fitxer de log
   * @param string | array | object  $var
   * @param boolean $date si volem guardar la data en ms
   * @param boolean $exit si volem parar l'execució
   */
   public static function log( $var,$date=false,$exit = false )
   {

      $date =($date)?time():'';
          if(!is_dir(APP. DS . 'logs' ))mkdir (APP. DS . 'logs' , 0777);//TODO pot haver
problemes al intentar crear el directori en servers Linux
```

```
      $fp = fopen(    APP. DS . 'logs' . DS .'log'.$date.'.log', 'a'    );
      fputs($fp,"[".date("Y-m-d H:i:s")."]");
      if( is_array( $var ) || is_object( $var ) )
      {
         fputs($fp, print_r( $var, true ));
      }
      elseif( is_string( $var ) )
      {
         fputs($fp,"string(" . strlen( $var ) . ") \"" .  $var  . "\"\n");
      }
      else
      {
         fputs($fp,var_dump( $var ));
      }
      fclose($fp);
      if( $exit )exit;
   }
  /**
   * Ens mostra per pantalla el valor de $var
   * @param string | array | object  $var
   * @param boolean $color el color amb el que es mostrarà el debug
   * @param boolean $exit si volem parar l'execució
   */
   public static function deb( $var, $color = "red", $exit = false )
   {

      echo "\n<div style=\"color:" . $color . "\">";
      echo "<hr />";
      echo "\n<pre>";

      if( is_array( $var ) || is_object( $var ) )
      {
         echo print_r( $var, true );
      }
      elseif( is_string( $var ) )
      {
         echo "string(" . strlen( $var ) . ") \"" . $var  . "\"\n";
      }
      else
      {
         var_dump( $var );
      }

      echo "</pre>";
      echo "<hr />";
      echo "</div>";

      if( $exit )
      {
         exit;
      }
   }
```

Altres mètodes

A part dels Helpers tenim a la nostra disposició altres mètodes que ens permeten agilitzar el procés de codificació que passem a explicar de forma resumida:

- **addCss(), addJs() i min():** Són mètodes que ens permeten incloure arxius css, js i css i js alhora a la pàgina.
- **model(), controller() i library()** permeten automàticament instanciar models, controladors i llibreries externes passats com a paràmetre.

Model

Totes les classes que fan de model dins el patró MVC hereten directament de la classe **ModelBase**. Aquesta classe, el que fa és cridar en el constructor a la classe SPDO que alhora, extén de PDO però li aplica el patró Singleton per a fer la connexió persistent.

ModeBase ens ofereix una API molt més senzilla i entenedora que treballant directament amb PDO per complir un dels requisits del Framework, reduir els temps per a la codificació de l'aplicació.

Els mètodes més destacats, que no els únics, d'aquesta classe els enumerem a continuació:

- **prepararVariable($nom,$variable)** Que permet protegir les dades que es passaran a la base de dades.
- **novaConsulta($sql)** Que és l'encarregada de llançar les consultes a la base de dades.
- **showQuery($sql)** Que ens permet veure per pantalla per així depurar la consulta llançada.

12.2 Procés d'execució

Per acabar aquest apartat farem un ràpid recorregut que realitza la nostra aplicació des del moment que rep una consulta d'una URL fins que mostra per pantalla la pàgina sol·licitada.

Suposem que ens hem registrat com a administrador i sol·licitem des del navegador la pàgina http://www.alec6.com/xmns/admin/editar-usuari/4.html

El fitxer .**htaccess** envia la petició cap a la pàgina **index.php** i s'executa el constructor de la classe **MVC**.

Aquest rep com a paràmetre get url i com a valor **admin/editar-usuari/4** que tracta les dades per a que cridi al mètode **editarUsuari($id_usuari)** del controlador **AdminController** passant-li com a paràmetre el valor **4.**

A continuació mostrem aquest mètode i l'expliquem pas a pas:

1. Mètode editarUsari($id_usuari) del controlador AdminController

```
2.  public function editarUsuari($id_usuari){
3.
4.      $model_users = $this->model('usuaris');
5.      $model_rols = $this->model('rols');
6.      $model_usuaris_classe = $this->model('usuarisClasse');
7.      $model_classe = $this->model('classes');
8.
9.      $this->variables['rols'] = $model_rols->getRols($id_usuari);
10.     $this->variables['usuari_editar'] = $model_users->getUserById($id_usuari);
11.         $this->variables['usuari_editar']->rol = $model_rols->getRolById($this->variables['usuari_editar']->id_rol)->rol;
12.     $this->variables['title']="Editar ". $this->variables['usuari_editar']->usuari;
13.
14.     switch($this->variables['usuari_editar']->rol){
15.        case 'alumne':
16.           $this->variables['classe'] = '';
17.           $classe = $model_usuaris_classe->getClasseByIdUser($id_usuari);
18.           if($classe){
19.              $id_classe = $classe[0]->id_classe;
20.                                  $this->variables['classe'] = $model_classe->getClasseById($id_classe)->nom;
21.           }
22.
23.           break;
24.        case 'professor':
25.           $this->variables['classe'] = '';
26.           $this->variables['alumnes'] = '';
27.           $classe = $model_classe->getClasseByIdProfessor($id_usuari);
28.           if($classe){
29.              $this->variables['classe'] = $classe[0]->nom;
30.                                  $ids_alumnes = $model_usuaris_classe->getUsersByIdClasse($classe[0]->id);
31.              $array_nom_alumnes = array();
32.              if($ids_alumnes){
33.                 foreach($ids_alumnes as $id_alumne){
34.                                  $alumne=$model_users->getUserById($id_alumne->id_usuari);
35.                    $array_nom_alumnes[]=$alumne->usuari;
36.                 }
37.              }
38.              $this->variables['alumnes'] = $array_nom_alumnes;
39.           }
40.           break;
41.     }
42.     $variables_menu['actual']='usuaris';
43.              $this->variables['menu'] = $this->view->getView('admin/menu', $variables_menu);
44.
```

```
45.
46.         $this->view->showView('admin/editar-usuari',$this->variables);
47.
48.     }
```

- Des de la línia 4 a la 7 s'instancien els models necessaris que necessitarem en aquest model.
- Des de la línia 9 a la 12 guardem valors a algunes de les variables que necessitarem posteriorment obtenint-les dels models.
- En la línia 14, a partir de l'id de rol de l'usuari amb id 4 decidim si es tracta d'un usuari alumne o d'un usuari professor, en el nostre cas és un usuari alumne.
- Mirem si l'alumne pertany a una classe, i si és així guardem el nom de la classe en una variable que posteriorment passarem a la vista altrament deixem el valor d'aquesta variable buida.
- Les línia 42 guarda una variable necessària per a la vista del menú principal on indica quina és la secció on ens trobem; en la 43, mitjançant el mètode **getView()** de la classe **View** encarregada de tractar les vistes obtenim la vista que fa de menú.
- Per finalitzar, en la línea 46, mitjançant el mètode **showView()** de la classe **View** mostrem per pantalla la vista passada com a primer paràmetre amb l'array de les variables necessàries en aquesta vista.

  Cal esmentar que per poder obtenir de l'array associatiu passat com a paràmetre els índex com a nom de variables i els seus valors fem servir la funció **extract()** de PHP encarregada de realitzar aquesta acció.

  Exemple:

Exemple de funcionament de la funció extract()

```
$variables['titol']= 'usuaris';$variables['nom']= 'Alexis';
extract($variables);
//Ara obtindriem el següent
$titol = 'usuaris';
$nom = 'Alexis';
```

13 Implantació i resultats

En aquest apartat mostrarem els resultats de la implantació de **XMNS** mostrant les interfícies principals.

Per començar s'explica la pàgina inicial de l'aplicació que permet l'accés i registre.

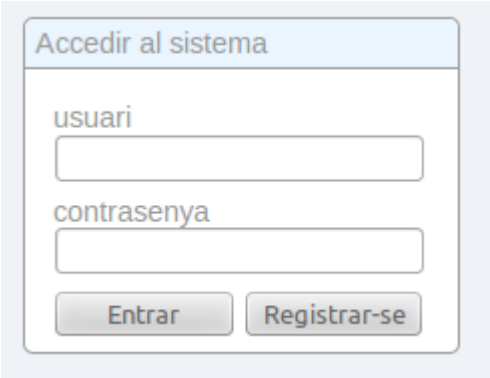

Figura 7

Des d'aquesta es permet accedir si ja ets un usuari registrat o registrar-se per a crear un nou usuari.

Si s'intenta accedir, es comprova que es tracta d'un usuari ja existent. Si no està registrat, es notifica mitjançant un missatge.

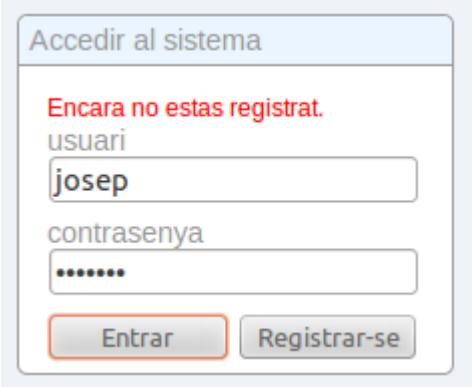

Figura 8

D'altra banda, si s'intenta registrar-se es comprova, primer que el camp usuari no està buit i que té mínim 3 caràcters.

Que el camp contrasenya tampoc està buit i que té mínim 6 caràcters.

Posteriorment també comprova que un usuari amb aquest nom no existeix en el sistema.

Si tot és correcte, permet l'accés a l'aplicació mostrant la pàgina d'aterratge corresponent al rol que té adjudicat.

Comprovacions d'aquest tipus es realitzen en tots els formularis de l'aplicació.

Com ja s'ha comentat anteriorment hi ha 4 rols distints. **usuari,alumne, professor i administrador**

Començarem veient el més senzill, el rol **usuari**:

A continuació mostrem la seva pàgina d'aterratge:

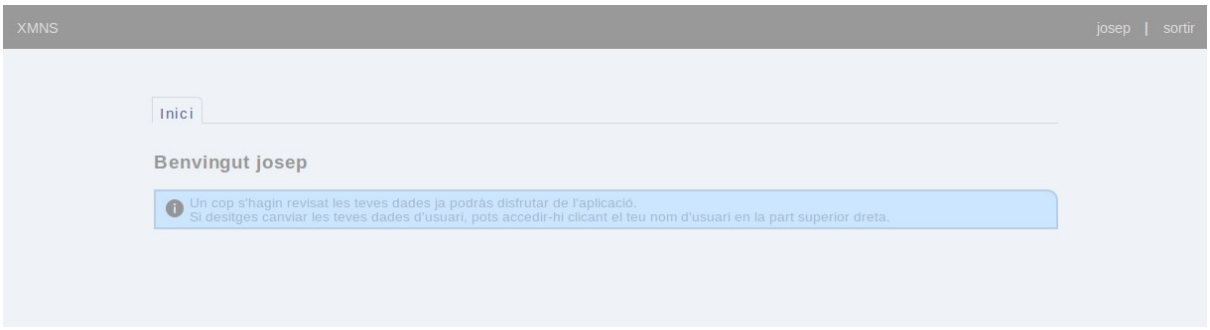

Figura 9

Aprofitem aquest punt per explicar detalls comuns de les interfícies de tots els rols:

- El logotip de la part superior esquerre sempre ens serveix per a retornar a la pàgina d'inici de interfície de l'usuari.
- A la part superior dreta tenim un petit menú amb 2 elements: el **nom de l'usuari** que ens porta a la pàgina d'edició de les dades de l'usuari (nom i contrasenya) i el botó de **sortir** per a sortir de l'aplicació i tornar al formulari d'accés i registre.
- Llavors tenim un menú horitzontal ,és el menú principal i varia depenent del rol que es té adjudicat i senyalant en quina secció ens trobem en aquest moment.
- Posteriorment el **títol** de la pàgina.
- Llavors se'ns mostra el contingut de la pàgina on ens trobem que, en la pàgina d'aterratge de l'usuari, se'ns mostren **dades interessants** i **notificacions**.

  Les notificacions són elements comunicatius que envia l'aplicació automàticament. La diferència entre un missatge i una notificació és: la notificació no té un remitent i, a més a més, es pot decidir deixar de ser visibles, en canvi, el missatge sempre es podrà tornar a revisar tot i que es marqui com a missatge llegit.

  Però això ho veurem en més detall quan explirquem l'enviament de missatges tant de part de alumnes com de part de professors.

Continuem explicant el rol **administrador,** que serà l'encarregat de donar rols a la resta d'usuaris i gestionar les classes ,és a dir, és qui crearà les classes i les hi assignarà professors i alumnes

A continuació mostrem la seva pàgina d'aterratge:

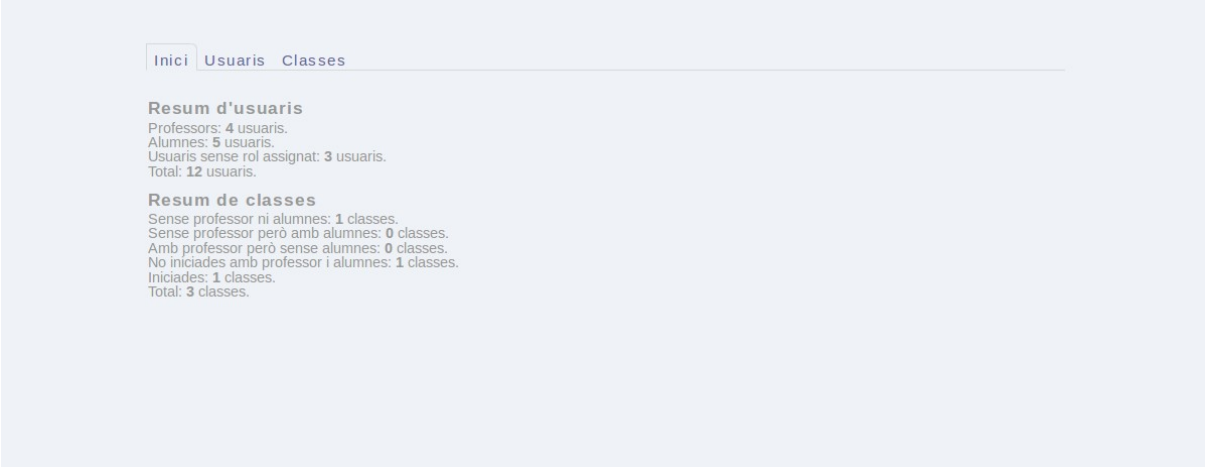

Figura 10

- Com podem veure, en la plana d'aterratge de l'administrador, se'ns mostra, de manera molt general, certes dades interessants d'usuaris registrats al sistema i de les classes creades en un moment determinat.
- El seu menú principal ens ofereix enllaços per a poder gestionar els usuaris i les classes.

Passem a veure la gestió d'usuaris:

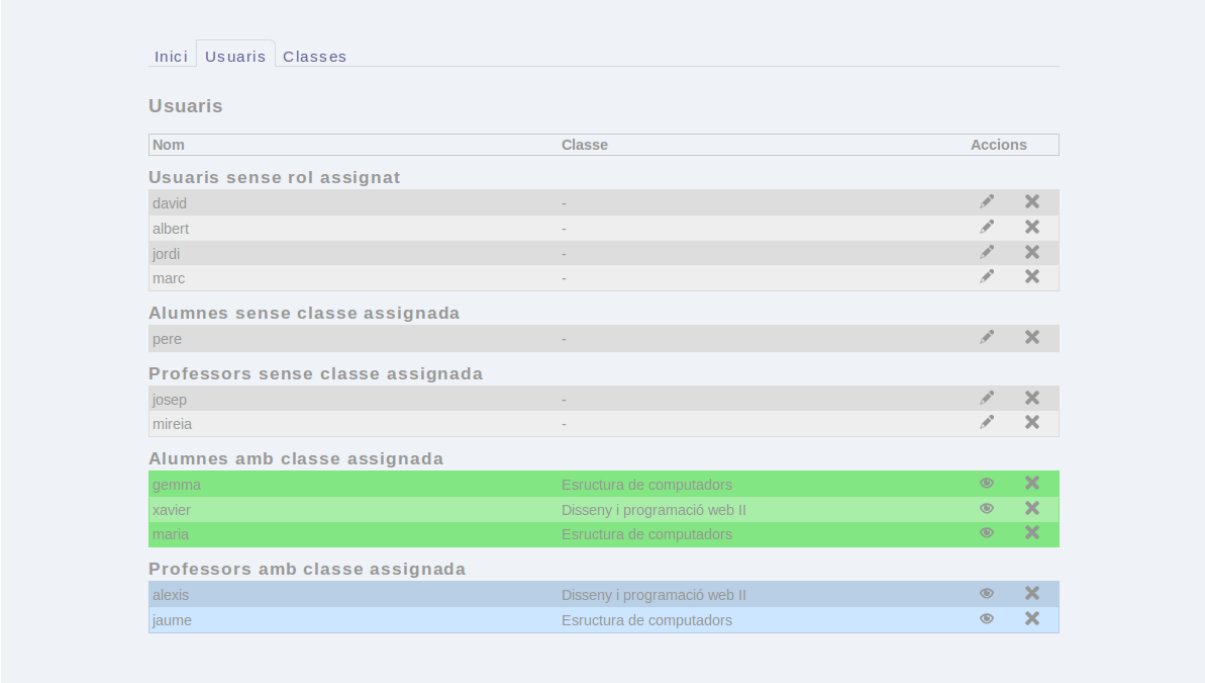

Figura 11

En aquesta pàgina l'administrador pot veure els usuaris que hi ha actualment registrats en l'aplicació.

Cal destacar que des d'aquesta pàgina només es poden canviar el rol assignat a un usuari a aquells

que encara no tenen assignada cap classe. Dels alumnes amb classe assignada es pot comprovar el seu rol i la classe que té assignada, i dels professors amb classe assignada en pot visualitzar a més els alumnes de la seva classe.

És des d'aquesta pàgina des d'on es poden eliminar els usuaris però sent conscient del que significa realitzar aquesta acció.

Ara passem a veure la gestió de classes:

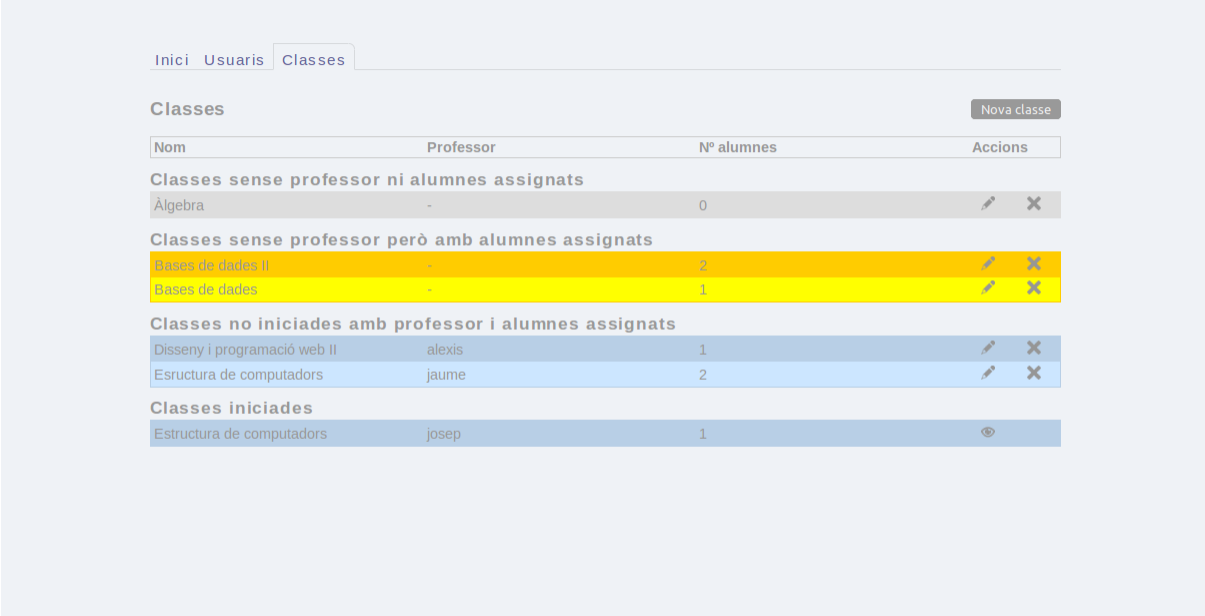

Figura 12

Com podem veure totes les pàgines amb dades tabulades tenen el mateix aspecte.

Fixem-nos que totes tenen les mateixes accions a realitzar, a excepció de les que es consideren Classes iniciades, que són les classes que ja tenen almenys un examen en el període on els alumnes han de respondre'l.

Des d'aquesta pàgina és també des d'on podem crear noves classes.

A continuació mostrem la pàgina de creació i edició de classes:

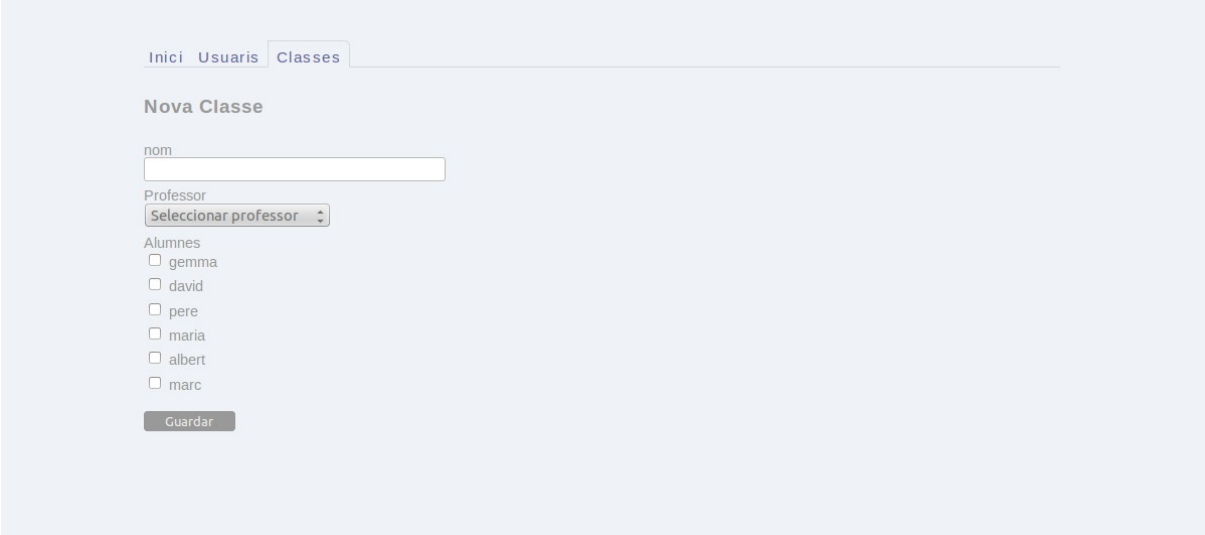

Figura 13

Aquesta pàgina té varis punts a comentar:

- Hem de definir com a mínim un **nom de classe** i que aquest no existeixi ja en el sistema.
- Els professors possibles són usuaris amb rol professor que encara no tenen classe assignada o usuaris sense rol assignat.
- Si es vol deixar la classe sense professor hem d'escollir la opció del desplegable professor "**Deixar sense professor**"
- Els possibles alumnes de la classe són alumnes sense classe assignada o usuaris sense rol assignat.
- Si es selecciona un professor que era un usuari sense rol, llavors automàticament queda la opció de seleccionar-lo com alumne deshabilitada; i si es selecciona com alumne un usuari sense rol, ens desapareix la opció d'escollir-lo com a professor.

Es poden canviar els participants d'una classe ja existent:

- Podem canviar el nom de la classe.
- Podem canviar el professor de la classe.
- Podem canviar els alumnes de la classe.

Cal destacar que els canvis que es realitzin en una classe seran notificats als usuaris participants a la classe, tant als antics membres com als nous.

Ara passem a veure el **rol de professor,** que és l'encarregat de gestionar la classe a la que ha estat

assignada, revisar l'evolució de la classe i crear exàmens per als alumnes d'aquesta classe.

A més a més, podrà enviar un missatge als alumnes que esculli de la seva classe.

Comencem amb la secció dels alumnes:

Se'ns mostrarà un llistat dels alumnes de la classe amb la opció per a cada alumne de veure les seves estadístiques:

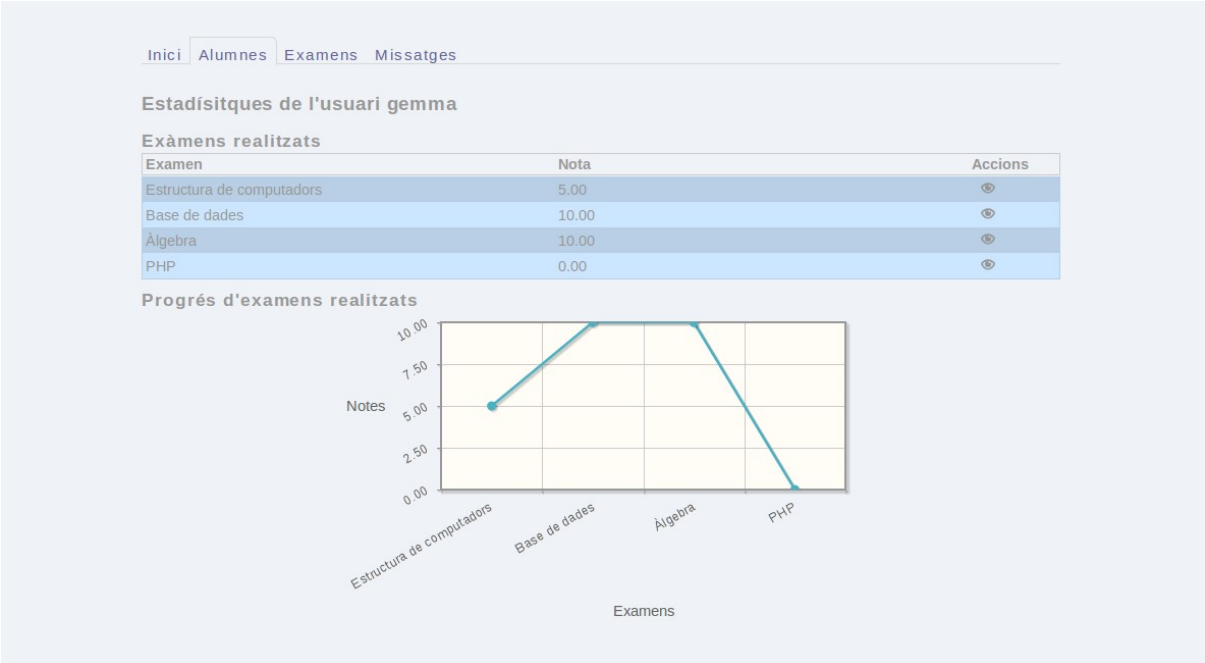

Figura 14

Des d'aquesta pàgina tenim la llista dels exàmens que ha realitzat l'alumne (o que no ha realitzat durant el temps establert per a que es realitzi) i la nota que ha obtingut **calculada de forma automàtica**.

A la part inferior es pot veure aquestes mateixes dades però sobre d'un gràfic lineal.

Continuem ara veient com un professor veu un examen realitzat per un alumne:

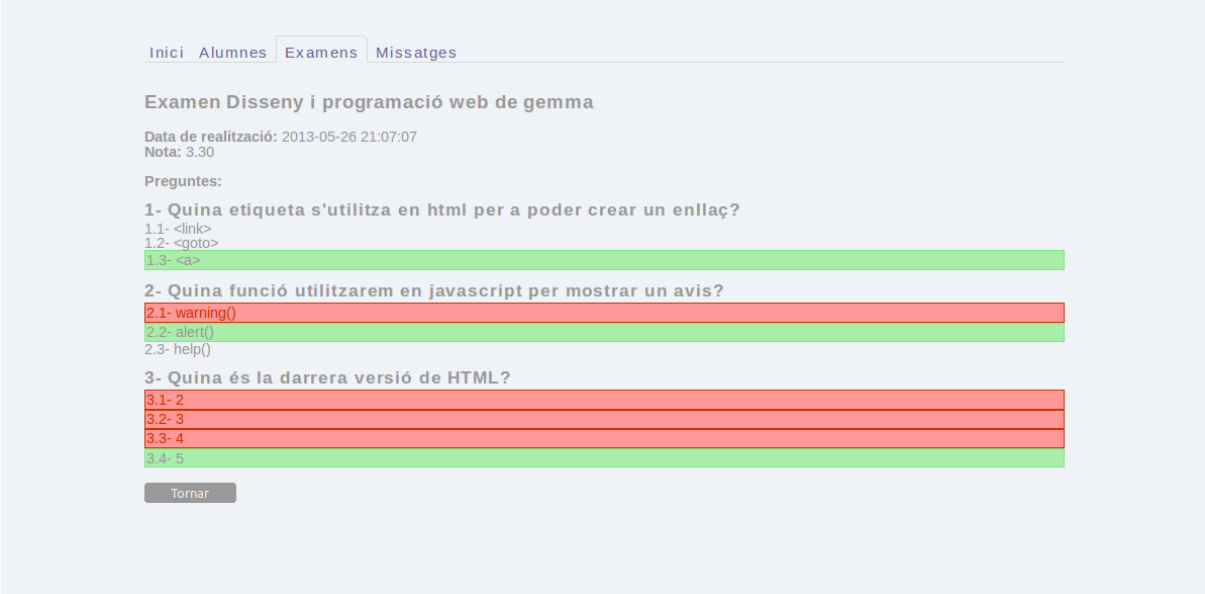

Figura 15

En l'exemple anterior es mostra la data en que l'alumne va realitzar l'examen. Si no l'hagués realitzat ens indicaria “examen no realitzat”.

Seguidament s'indica la nota.

Llavors les respostes de l'examen amb el següent criteri:

- Si l'alumne ha encertat la resposta aquesta queda marcada en verd.
- Si l'alumne ha errat en la resposta, la resposta escollida per l'usuari queda marcada en vermell i la resposta correcte en verd.
- Si l'alumne no ha contestat cap resposta, totes les respostes queden marcades en vermell a excepció de la resposta correcte.

Ara passem a veure la secció de exàmens:

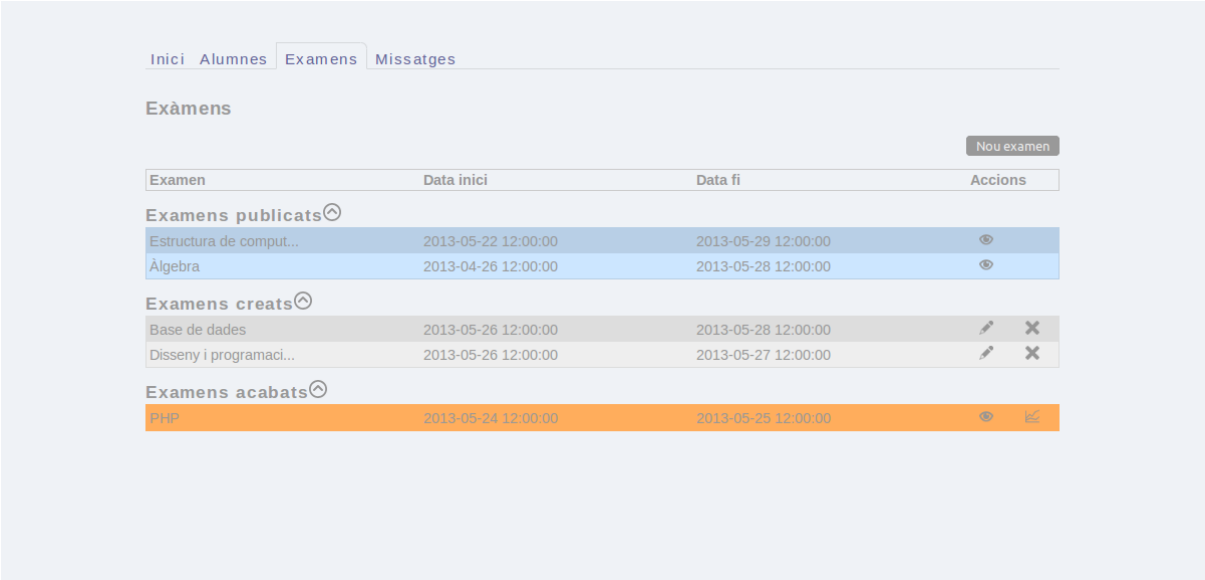

Figura 16

En aquesta pàgina, els exàmens estan dividits en:

- **Publicats**: És a dir que estem entre les dates en que els alumnes poden resoldre l'examen.

  Aquests tipus d'exàmens permet veure'n les dades de l'examen.

- **Creats**: És a dir exàmens que la data actual encara no ha arribat al període on els alumnes poden respondre'l.

  Aquests tipus d'exàmens es permet editar-lo i eliminar-lo.

- **Acabats**:És a dir exàmens que el seu període de resolució ja ha expirat.

  Aquests tipus d'exàmens es permet veure'n les seves dades i les estadístiques obtingudes per la resolució d'aquest per part dels alumnes.

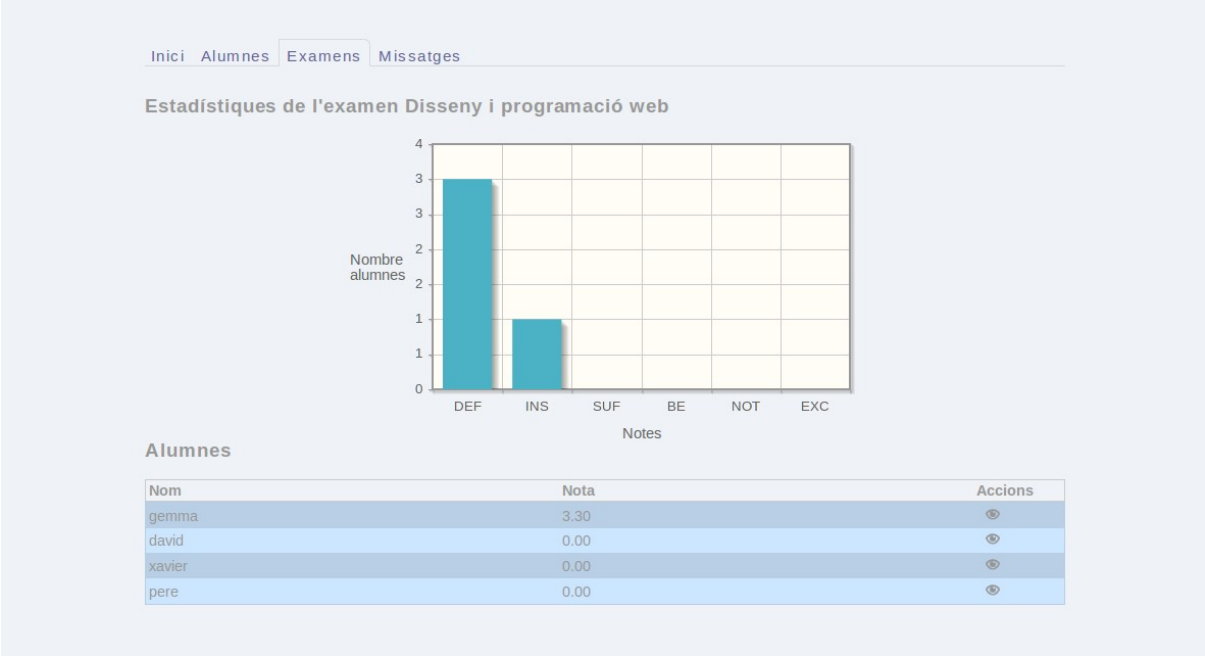

Figura 17

Veiem que les notes que s'obtenen de l'examen es mostren en un gràfic de barres i que les notes estan presentades com antigament es feia:

- **DEF**: de 0 a 4.
- **INS**:de 4 a 5.
- **SUF**:de 5 a 6:
- **BE**:de 6 a 7.
- **NOT**:De 7 a 9.
- **EXC**:De 9 a 10.

Passem a veure com un professor pot crear un examen:

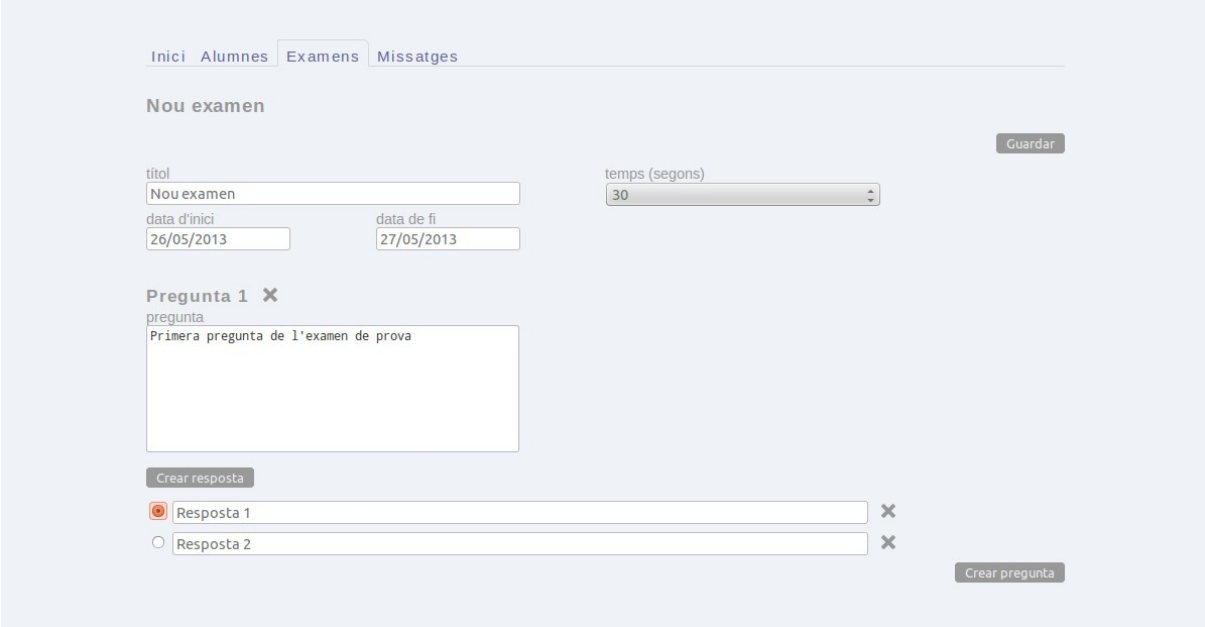

Figura 18

Anem a explicar detalladament aquesta pàgina:

- Hem d'escriure obligatòriament un nom d'examen únic dins d'aquesta classe.
- La data d'inici, ha de ser com a mínim la data actual.
- La data final ha de ser almenys 1 dia posterior a la data d'inici.
- Al seleccionar una data es considera que serà les 12 del migdia del dia escollit.
- El temps de realització serà només de 30 60 o 120 segons.
- Per crear preguntes hem de clicar al botó corresponent tants cops com preguntes vulguem.
- Un cop escrita la pregunta hem de crear tantes respostes com vulguem, escriure-les i marcar la resposta correcte.
- Un cop revisades les dades cliquem al botó de guardar situat a la part superior dreta.

Ara anem a explicar la secció de missatges:

Els missatges poden tenir 3 estats:

- Rebuts i llegits.
- Rebuts però no llegits
- Enviats.

Ara veiem la pantalla per a crear un missatge:

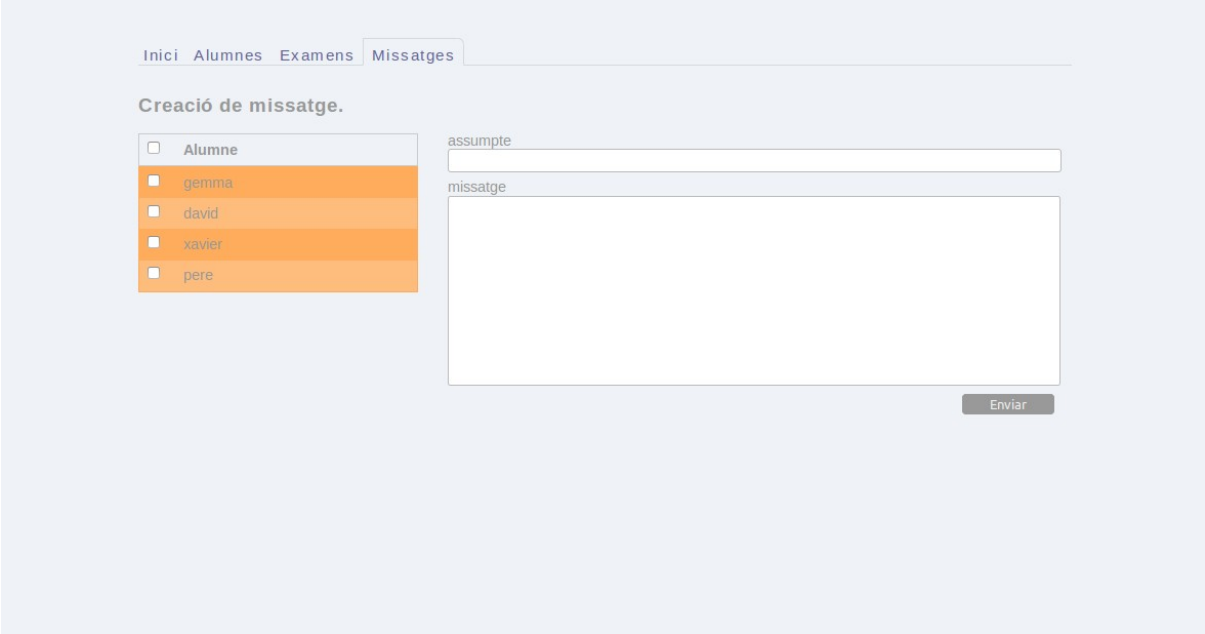

Figura 19

Com a mínim un receptor, un assumpte i un missatge són necessaris per a poder enviar un missatge

Per finalitzar, anem a veure el **rol alumne**:

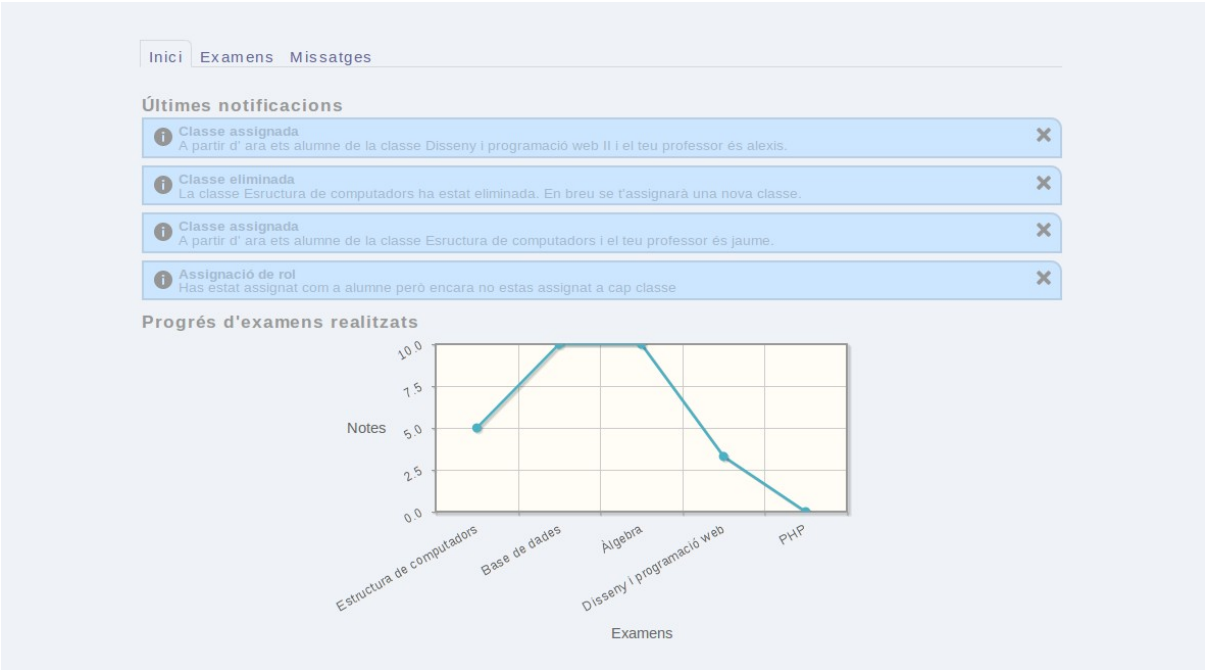

Figura 20

Com veiem en l'anterior imatge, les dades interessants que pot veure un alumne en la seva pàgina d'aterratge és la seva evolució obtinguda a partir dels exàmens realitzats plasmada en un gràfic lineal.

Passem a comentar la secció de missatges:

Al igual que per a un professor disposem de dues pantalles per a la gestió de missatges:

- **Llista de missatges**: Que és idèntica a la pantalla que té un professor.
- **Crear un missatge**: Que és igual a la d'un professor però amb la diferència de que no pot seleccionar el receptor del missatge ja que aquest és sempre el seu professor; el nostre sistema té restringit l'enviament de missatges per part de un alumne cap al seu professor.

Ja per acabar veiem la secció d'exàmens:

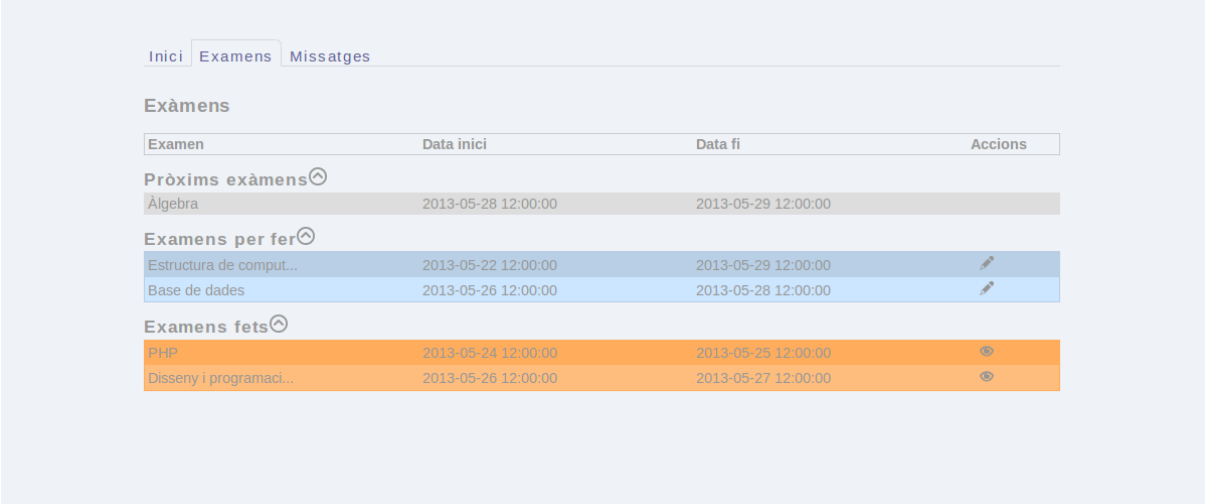

Figura 21

Tenim 3 tipus d'exàmens:

- **Pròxims exàmens**: Són aquells els quals el període de realització encara no ha arribat.
- **Exàmens fets**: Els exàmens que el seu període de realització ja ha expirat, s'hagi o no fet.

  L'acció que podem fer d'aquest examen és visualitzar-lo amb els mateixos criteris que té un professor quan consulta un examen realitzat per un alumne i que anteriorment ja hem explicat.

- **Exàmens per fer**: Aquells en els que el període de realització és en aquell moment.

  Les accions que podem fer és evidentment la seva realització i que passem a veure tot seguit:

  Quan seleccionem l'acció de realitzar un examen per fer se'ns mostra la següent pàgina:

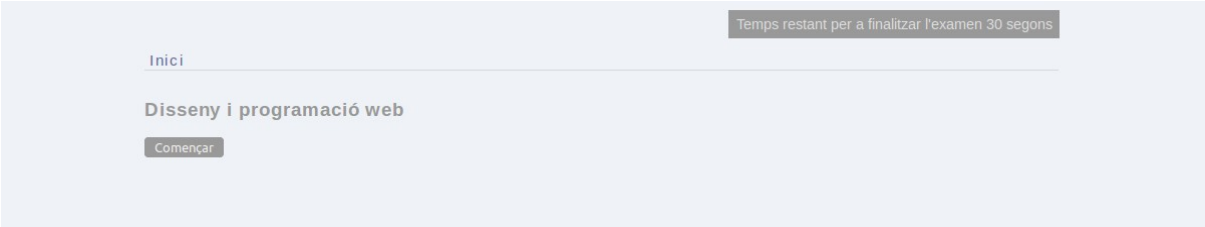

Figura 22

on podem veure, el nom de l'examen i el temps que tenim per a realitzar l'examen.

Al clicar en el botó començar se'ns mostren les preguntes a respondre corresponents a l'examen que s'està realitzant i el temporitzador va restant.

Figura 23

Algun punt a destacar d'aquesta interfície:

- Es pot decidir que la prova està finalitzada abans que acabi el temps.
- Si el temps s'esgota, les preguntes no contestades queden com a preguntes errades i es dona com a examen finalitzat.
- Quan es finalitza un examen se'ns mostra un resum amb la data de realització, la nota obtinguda i la resolució de la prova amb el criteri explicat en la visualització d'exàmens.

14 Conclusions

XMNS partia d'una premissa que consistia en realitzar una aplicació web per a la correcció automàtica de proves tipus test.

Després de realitzar un petit treball de camp per a cercar aplicacions semblants ja existents i amb la experiència personal pròpia, es va decidir afegir més funcionalitats que simplement corregir exàmens.

Bàsicament es va afegir un sistema de comunicació entre docent i alumnes i una manera de visualitzar les dades obtingudes a partir dels resultats de les proves útils pel seguiment del procés formatiu tant per part del professor com dels docents. Tot això dins un entorn on el procés d'aprenentatge d'ús sigués mínim i senzill.

La finalitat d'aquests valors afegits era obtenir un entorn més real i semblant a una classe presencial, cosa que en les aplicacions analitzades o era massa complexa el seu ús o massa simple i limitat.

La valoració objectiva és que els objectius proposats s'han complert de manera satisfactòria i dins els temps adjudicats.

Les desviacions de la planificació inicial han estat mínimes, tan sols algunes relacionades en la manera de mostrar certes dades que es van detectar en el moment de codificació, i la més important la temporització en l'estat final que es va haver d'ajustar degut a circumstàncies externes.

Totes resoltes de manera satisfactòria.

Com s'ha comentat en aquest document, aquest treball no pretenia ser un treball final sinó més aviat un prototip, una manera de plasmar un concepte. Apropar més les classes en entorns virtuals a les classes en entorns presencials.

Per tant, crec que els resultats finals han estat reeixits, ja que apropa més professors i alumnes en un entorn virtual.

15 Treball futur

Finalitzat el treball és interessant comentar certes millores que es podrien implementar. Algunes ja eren conegudes abans d'iniciar el treball, per exemple:

- Afegir un sistema de seguretat més eficient en el moment de registrar-se, ja sigui mitjançant un sistema de captchas o mitjançant sistemes d'acceptació a partir d'un enllaç dins d'un correu electrònic.
- Permetre que un professor ho sigui de més d'una classe.
- Permetre que un alumne ho sigui de més d'una classe.
- Afegir assignatures a una classe.
- Permetre adjuntar fitxers en els missatges enviats.

Altres varen sorgir quan ja s'havia iniciat el treball com per exemple:

- Millores a nivell de codi en el Framework utilitzat, com per exemple:
    - Millorar la jerarquització de les vistes.
    - Sistema de caché.
- Creació d'una aplicació per a smartphones, com a mínim per a la part d'alumnes, i així puguin respondre les proves i visualitzar els missatges.
- Fer més visible quan un usuari té un nou missatge o si un alumne té exàmens a resoldre.

16 Webgrafia

Hot Potatoes http://hotpot.uvic.ca

Moodle https://moodle.org

TestGip http://personales.upv.es/arodrigu/testgip

Daypo http://www.daypo.com

w3schools http://www.w3schools.com

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

phpMyAdmin http://es.wikipedia.org/wiki/PhpMyAdmin

MysqlWorkbench http://es.wikipedia.org/wiki/MySQL_Workbench

umbrello http://es.wikipedia.org/wiki/Umbrello

PHP http://www.php.net/

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

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

JQuery http://jquery.com/

jqPlot http://www.jqplot.com/

Model en cascada http://es.wikipedia.org/wiki/Desarrollo_en_cascada

UML https://es.wikipedia.org/wiki/Lenguaje_Unificado_de_Modelado

Índex de figures