Progetto di Tecnologie Informatiche per il web Anno 2020 / 2021

Versione RIA

Professore Piero Fraternali

Alessandro Mazza

Cod. Persona 10670112 Matricola 908651

Davide Marinotto

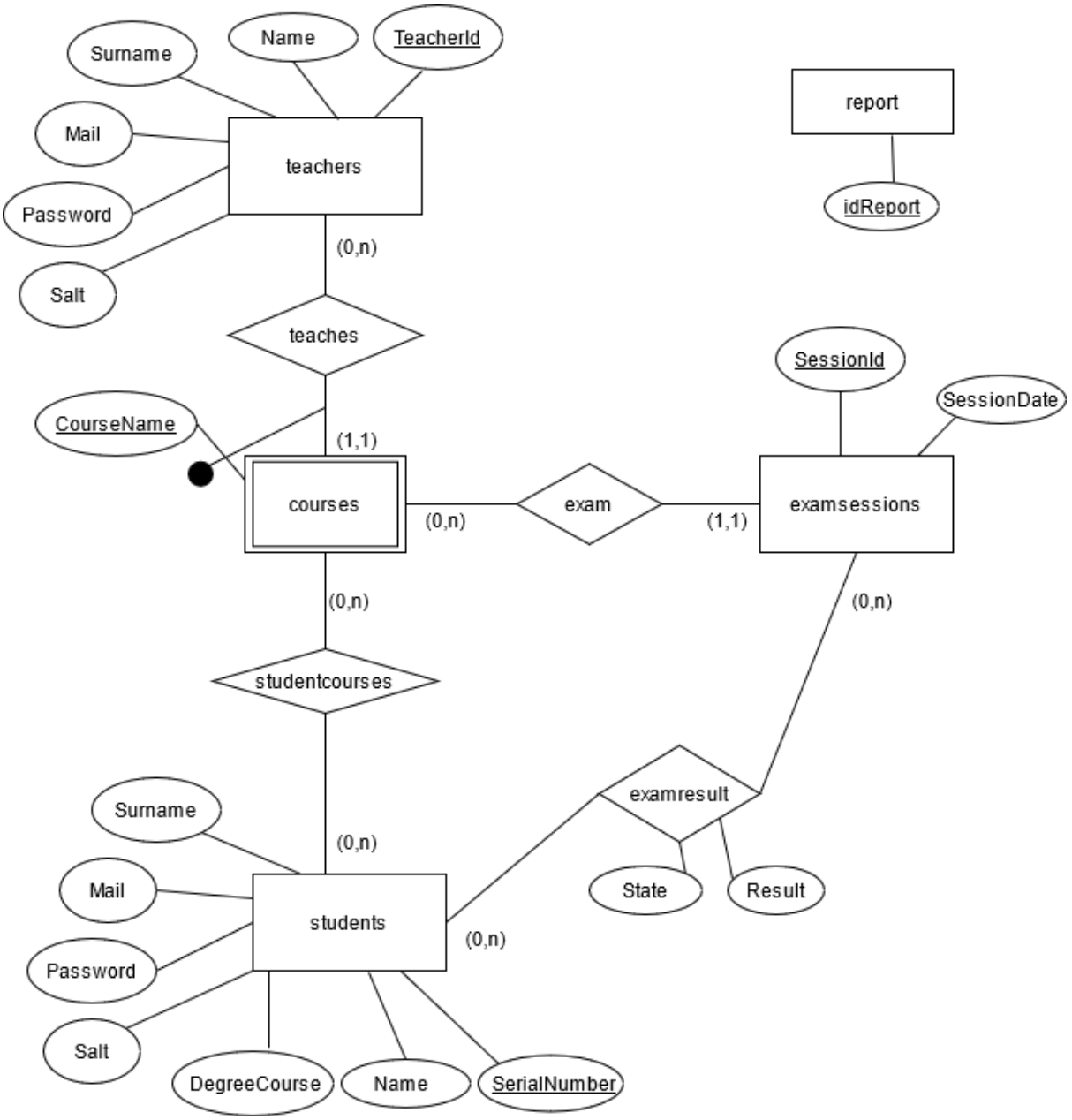
Cod. Persona 10675966 Matricola 909291

Analisi dei Dati

Un'applicazione permette di verbalizzare gli **esiti degli esami** di un appello.
Il **docente** accede tramite login e seleziona nella HOME page un corso da una lista dei propri **corsi** ordinata in modo alfabetico decrescente e poi una **data** d'**appello del corso** scelto selezionata da un elenco ordinato per data decrescente. Ogni corso ha un solo docente.
La selezione dell'appello porta a una pagina ISCRITTI, che mostra una tabella con tutti gli iscritti all'appello. La tabella riporta i seguenti dati: **matricola, cognome e nome, email, corso di laurea, voto e stato di valutazione**. Il voto può non essere ancora definito.
Lo stato di valutazione dello studente rispetto all'appello può assumere i valori: non inserito, inserito, pubblicato, rifiutato e verbalizzato. Selezionando un'etichetta nell'intestazione della tabella, l'utente ordina le righe in base al valore di tale etichetta (ad esempio, selezionando "cognome" la tabella è riordinata in base al cognome). Successive selezioni della stessa etichetta invertono l'ordinamento: si parte con l'ordinamento crescente. Il valore del voto viene considerato ordinato nel modo seguente: , assente, rimandato, riprovato, 18, 19, …, 30, 30 e lode. Ad ogni riga corrisponde un bottone "MODIFICA". Premendo il bottone compare una pagina con una form che mostra tutti i dati dello **studente** selezionato e un campo di input in cui è possibile scegliere il voto. L'invio della form provoca la modifica o l'inserimento del voto. Inizialmente le righe sono nello stato di valutazione "non inserito". L'inserimento e le successive eventuali modifiche portano la riga nello stato di valutazione "inserito". Alla tabella è associato un bottone PUBBLICA che comporta la pubblicazione delle righe con lo stato di valutazione INSERITO. La pubblicazione rende il voto non più modificabile dal docente e visibile allo studente e cambia lo stato di valutazione della riga dello studente a "pubblicato". Lo studente accede tramite login e seleziona nella HOME page un **corso tra quelli a cui è iscritto** mediante una lista ordinata in modo alfabetico decrescente e poi una data d'appello del corso scelto selezionata da un elenco ordinato per data decrescente. Uno studente può iscriversi a più appelli dello stesso corso. La selezione della data d'appello porta a una pagina ESITO che mostra il messaggio "Voto non ancora definito" se il docente non ha ancora pubblicato il risultato per quello studente in quell'appello. Altrimenti, la pagina mostra i dati dello studente, del corso, dell'appello e il voto assegnato. Se il voto è tra 18 e 30 e lode compare un bottone RIFIUTA. Premendo tale bottone la pagina mostra gli stessi dati con la dizione aggiunta "Il voto è stato rifiutato" e senza il bottone RIFIUTA. Il rifiuto del voto cambia lo stato di valutazione a "rifiutato" della riga dello studente per quell'appello nella pagina ISCRITTI del docente. Nella pagina ISCRITTI del docente la tabella degli iscritti è associata anche a un bottone VERBALIZZA. La pressione del bottone provoca il cambio di stato a "verbalizzato" per le righe nello stato "pubblicato" o "rifiutato" e comporta anche la creazione di un verbale e la disabilitazione della possibilità di rifiutare il voto. Il rifiuto implica la verbalizzazione di "rimandato" come voto.
Un **verbale** ha: un **codice generato dal sistema**, una data e ora di creazione ed è associato all'appello del corso a cui si riferisce e agli studenti (con nome, cognome, matricola e voto) che passano allo stato "verbalizzato". A seguito della pressione del bottone VERBALIZZA compare una pagina VERBALE che mostra i dati completi del verbale creato.

Legenda: entità, attributi, relazioni

Schema ER


Surname
Name
TeacherId
Mail
teachers
Password
Salt
(0,n)
teaches
report
idReport
SessionId
SessionDate
CourseName
(1,1)
courses
(0,n)
exam
(1,1)
examsessions
(0,n)
(0,n)
studentcourses
examresult
Surname
(0,n)
Mail
students
(0,n)
State
Result
Password
Salt
DegreeCourse
Name
SerialNumber

Local Database Schema

- **teachers**
  ```
  CREATE TABLE `teachers` (
  `TeacherId` int NOT NULL,
  `Name` varchar(45) NOT NULL,
  `Surname` varchar(45) NOT NULL,
  `Mail` varchar(320) NOT NULL,
  `Password` varchar(64) NOT NULL,
  `Salt` varchar(12) NOT NULL,
  PRIMARY KEY (`TeacherId`)
   )
  ```
- **students**
  ```
  CREATE TABLE `students` (
    `SerialNumber` int NOT NULL,
    `Name` varchar(45) NOT NULL,
    `Surname` varchar(45) NOT NULL,
    `Mail` varchar(320) NOT NULL,
    `Password` varchar(64) NOT NULL,
    `DegreeCourse` varchar(45) NOT NULL,
    `Salt` varchar(12) NOT NULL,
    PRIMARY KEY (`SerialNumber`)
  )
  ```
- **studentcourses**
  ```
  CREATE TABLE `studentcourses` (
    `StudentSerialNumber` int NOT NULL,
    `CourseName` varchar(45) NOT NULL,
    PRIMARY KEY (`CourseName`,`StudentSerialNumber`)
  )
  ```
- **report**
  ```
  CREATE TABLE `report` (
    `idReport` int NOT NULL DEFAULT '0',
    PRIMARY KEY (`idReport`)
  )
  ```
- **examsessions**
  ```
  CREATE TABLE `examsessions` (
    `SessionId` int NOT NULL,
    `CourseName` varchar(45) NOT NULL,
    `SessionDate` date NOT NULL,
    PRIMARY KEY (`SessionId`)
  )
  ```

- **examresult**

```
CREATE TABLE `examresult` (
  `SessionId` int NOT NULL,
  `StudentSerialNumber` int NOT NULL,
  `State` varchar(45) NOT NULL DEFAULT 'NON INSERITO',
  `Result` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`SessionId`,`StudentSerialNumber`)
)
```

- **courses**

```
CREATE TABLE `courses` (
  `CourseName` varchar(45) NOT NULL,
  `TeacherId` int NOT NULL,
  PRIMARY KEY (`CourseName`,`TeacherId`)
)
```

Eventi

Suddividiamo gli eventi in base al contesto in cui possono essere invocati,definiamo quindi tre gruppi Eventi in Comune, Eventi Insegnante, Eventi Studente

Eventi in Comune

Login
- Verifica che le credenziali appartengono a un insegnante o a uno studente

Logout
- Cancella la sessione dell utente

Click su bottone Ritorna alla home
- Riporta alla pagina con l'elenco dei corsi

Click su bottone Stampa
- Stampa la pagina per un possibile utilizzo cartaceo o la necessità di generare pdf

Eventi Insegnante

Selezione **data appello** di un corso
- Mostra l'elenco degli iscritti a tale appello, e tutte le azioni che scatenano gli eventi successivi

Click su intestazione **elenco iscritti**
- Riordinamento

Click su bottone **Modifica**
- Assegna Valutazione a uno studente
- Imposta lo stato ad INSERITO

Click su bottone **Inserimento Multiplo**
- Assegna Valutazioni a più studenti
- Imposta gli stati ad INSERITO

Click su bottone **Close** (icona X)
- Chiudi finestra modale

Click su bottone **Pubblica**
- Pubblica le valutazioni registrate degli studenti iscritti
- Imposta lo stato a PUBBLICATO

Click su bottone **Verbalizza**
- Verbalizza le valutazioni registrate degli studenti iscritti
- Imposta lo stato a VERBALIZZATO

Eventi Studente

Selezione **data appello** di un corso
- Mostra la valutazione se disponibili se disponibile oppure un messaggio di cortesia

Click su bottone **Rifiuta**
- Verbalizza le valutazioni assegnata dall'insegnante
- Imposta lo stato a RIFIUTATO

Eventi & azioni

Client side		Server side	
Event	**Action**	**Event**	**Action**
index -> login form -> submit	Controllo correttezza formale Dati	POST username password	Controllo credenziali
Logout	-	GET	Fine Sessione
button.backToHome Btn -> click	Riporta alla pagina con l'elenco dei corsi	-	-
button.printPage -> click	Stampa la pagina	-	-
Home -> load	Aggiorna view con corsi i propri corsi e i rispettivi appelli	GET	Estrazione elenco corsi tenuti l'insegnante e i rispettivi appelli
Home -> elenco corsi -> seleziona appello	Aggiorna view con elenco iscritti all'appello	GET examSessionID	Estrazione elenco studenti iscritti all'appello examSessionID
Home -> elenco corsi -> seleziona appello-> click intestazione tabella iscritti	Aggiorna view con la tabella iscritti ordinata con il criterio scelto	-	-
Home -> elenco corsi -> elenco iscritti -> click modifica	Aggiorna view aprendo una modale con dati studente e campo selezione valutazione	GET studentId examSessionID	Estrazione i dati dello studente con matricola studentId iscritto all'appello examSessionID
Home -> elenco corsi -> elenco iscritti -> modifica -> invia -> submit	Aggiorna view con elenco iscritti all'appello	POST setResultSessionID setResultStudentID result	Inserimento della valutazione result allo studente setResultStudentID nell'appello setResultSessionID

Home -> elenco corsi -> elenco iscritti -> click Inserimento Multiplo	Aggiorna view aprendo una modale con i dati degli studenti e per ognuno un campo selezione valutazione	GET examSessionID	Estrazione i dati di tutti gli studente con stato 'NON INSERITO' iscritti all'appello examSessionID
Home -> elenco corsi -> elenco iscritti -> Inserimento Multiplo -> invia -> submit	Aggiorna view con elenco iscritti all'appello	POST examSessionID file json, array di { result: valutazione, student: matricola, }	Inserimento di ogni valutazione (result) per ogni studente (matricola) a esso associato) nella sessione (examSessionID)
Home -> elenco corsi -> elenco iscritti -> modifica -> close	Chiudi la finestra modale aperta, non serve ricaricare la view	-	-
Home -> elenco corsi -> elenco iscritti -> pubblica	Aggiorna view con elenco iscritti all'appello	POST examSessionId	Pubblica le valutazioni della sessione
Home -> elenco corsi -> elenco iscritti -> verbalizza	Aggiorna view con elenco iscritti all'appello	GET examSessionId	Verbalizza le valutazioni della sessione e restituisce la pagina del verbale
Home -> load	Aggiorna view con corsi i propri corsi e i rispettivi appelli	GET	Estrazione elenco corsi e appelli a cui lo studente è iscritto
Home -> elenco corsi -> seleziona appello	Aggiorna view con valutazione del appello	GET examSessionID	Estrazione dati studente riferiti all'appello identificato da examSessionID
Home -> elenco corsi -> valutazione -> rifiuta	Aggiorna view con il nuovo stato di valutazione del appello	POST sessionId refuseState	Rifiuta la valutazione assegnata all'appello sessionId

Controller / event handler

Client side		Server side	
Event	**Controller**	**Event**	**Controller**
index -> login form -> submit	Function makeCall	POST username password	LoginAction (servlet)
Logout	Function makeCall	GET	LogoutAction (servlet)
button.backToHome Btn -> click	Function PageOrchestrator	-	-
button.printPage -> click	Function PageOrchestrator	-	-
Home -> load	Function PageOrchestrator	GET	GetTeacherCourses (servlet)
Home -> elenco corsi -> seleziona appello	Function CorsiElenco.show	GET examSessionID	GetSubscribedStudent (servlet)
Home -> elenco corsi -> seleziona appello-> click intestazione tabella iscritti	sort	-	-
Home -> elenco corsi -> elenco iscritti -> click modifica	Function ModalModifyResult.show	GET studentId examSessionID	SetExamResult (servlet)
Home -> elenco corsi -> elenco iscritti -> modifica -> invia -> submit	Function makeCall	POST setResultSessionID setResultStudentID result	SetExamResult (servlet)
Home -> elenco corsi -> elenco iscritti -> click Inserimento Multiplo	Function ModalBulkModifyResults.show	GET examSessionID	SetBulkOfResult (servlet)

Home -> elenco corsi -> elenco iscritti -> Inserimento Multiplo -> invia -> submit	Function makeCall	POST
		examSessionID
		file json, array di
		{
		result: valutazione,
		student: matricola,
		}
			SetBulkOfResult (servlet)

Home -> elenco corsi -> elenco iscritti -> modifica -> close	Function ModalModifyResult.reset	-	-
Home -> elenco corsi -> elenco iscritti -> pubblica ->submit	Function IscrittiElenco.publishExam	POST examSessionId	GetSubscribedStudents (servlet)
Home -> elenco corsi -> elenco iscritti -> verbalizza -> submit	Function IscrittiElenco.verbalizeExam	GET examSessionId	Verbale (servlet)
Home -> load	Function makeCall	GET	GetStudentCourses (servlet)
Home -> elenco corsi -> seleziona appello	Function CorsiElenco.show	GET examSessionID	GetExamResult (servlet)
Home -> elenco corsi -> valutazione -> rifiuta	Function makeCall	POST sessionId refuseState	GetExamResult (servlet)

Server side: DAO & model objects

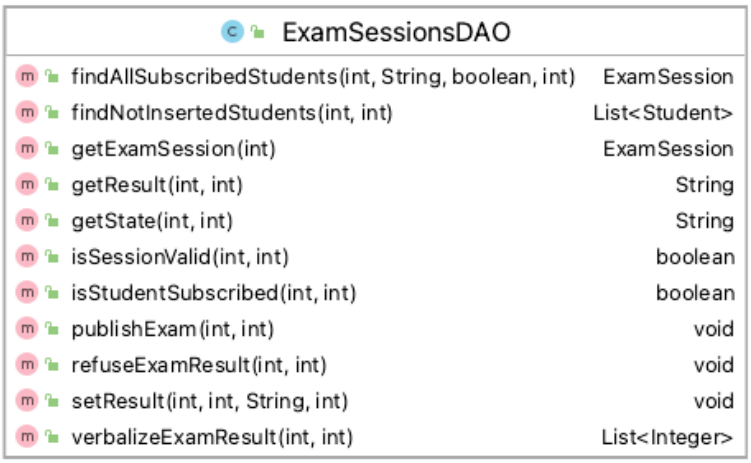
Controllers

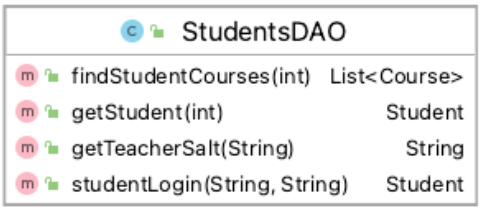
- GetExamResult
- GetStudentCourses
- GetSubscribedStudents
- GetTeacherCourses
- GetVerbale
- LoginAction
- LogoutAction
- SetBulkOfResults
- TeacherSetExamResult

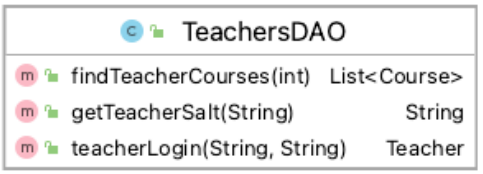
Model objects (Beans)

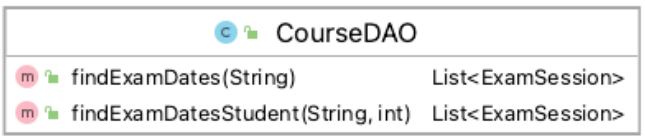
- Course
- ExamResult
- ExamSession
- Student
- Teacher
- Verbale

Data Access Objects (Classes)









Client side: view & view component

Index

Login Form

Teacher

- PageTitle
    - update() Aggiorna il contenuto del titolo della pagina
    - reset() Pulisce il contenuto del titolo della pagina
- CorsiElenco
    - show() richiede al server i dati dell'elenco corsi
    - update() riceve dati server e aggiorna l’elenco
    - reset() toglie la visibilità del contenitore dei corsi
- IscrittiElenco
    - publishExam() associa al componente le funzioni per richiedere al server una pubblicazione
    - verbalizeExam() associa al componente le funzioni per richiedere al server una verbalizzazione
    - show() richiede al server i dati dell'elenco corsi
    - update() riceve dati server e aggiorna l’elenco
    - reset() toglie la visibilità del contenitore dell elenco iscritti
- Verbale
    - show() richiede al server i dati del verbale
    - update() riceve dati server e genera lato html il verbale
    - reset() toglie la visibilità del verbale
- ModalModifyResult
    - setResult() associa al componente le funzioni per richiedere al server una modifica al voto di uno studente
    - show() richiede al server i dati del voto
    - update() riceve i dati dal server e modifica il voto
    - reset() toglie la visibilità alla pagina modale
- ModalBulkModifyResults
    - setResults() associa al componente le funzioni per richiedere al server una modifica al voto di più studenti
    - show() richiede al server i dati riguardanti tutti i voti che si possono modificare
    - update() riceve i dati dal server e compie la modifica multipla
    - reset() toglie la visibilità alla pagina modale

Student

- PageTitle
    - update() Aggiorna il contenuto del titolo della pagina
    - reset() Pulisce il contenuto del titolo della pagina
- CorsiElenco
    - show() richiede al server i dati dell'elenco corsi
    - update() riceve dati server e aggiorna l’elenco
    - reset() toglie la visibilità del contenitore dei corsi
- ResultPage
    - show() richiede al server i dati della valutazione
    - update() riceve dati server mostra la valutazione
    - reset() toglie la visibilità della valutazione