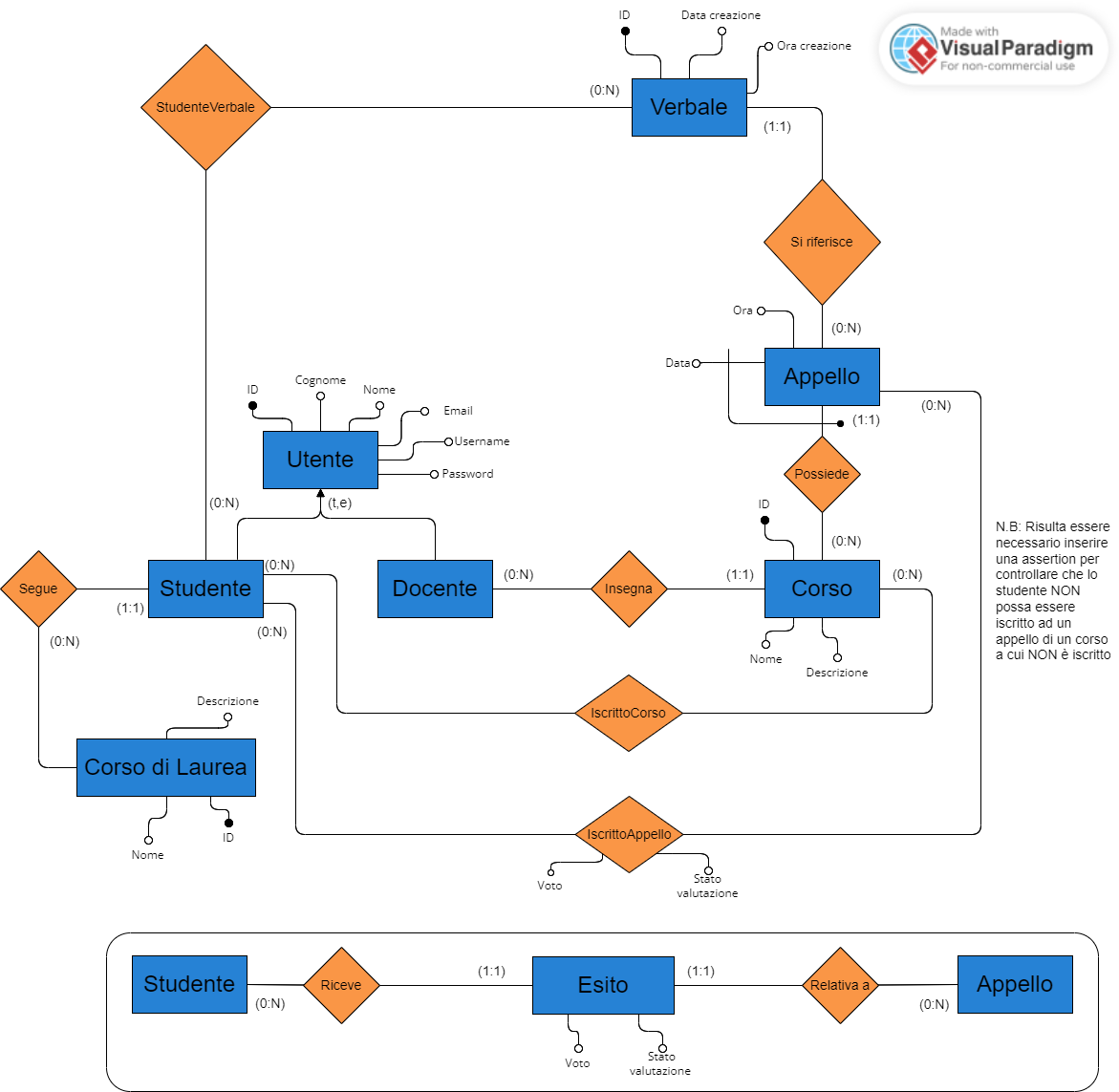
**Esercizio 4: verbalizzazione degli esami - Versione HTML pura**

**Analisi dei dati**

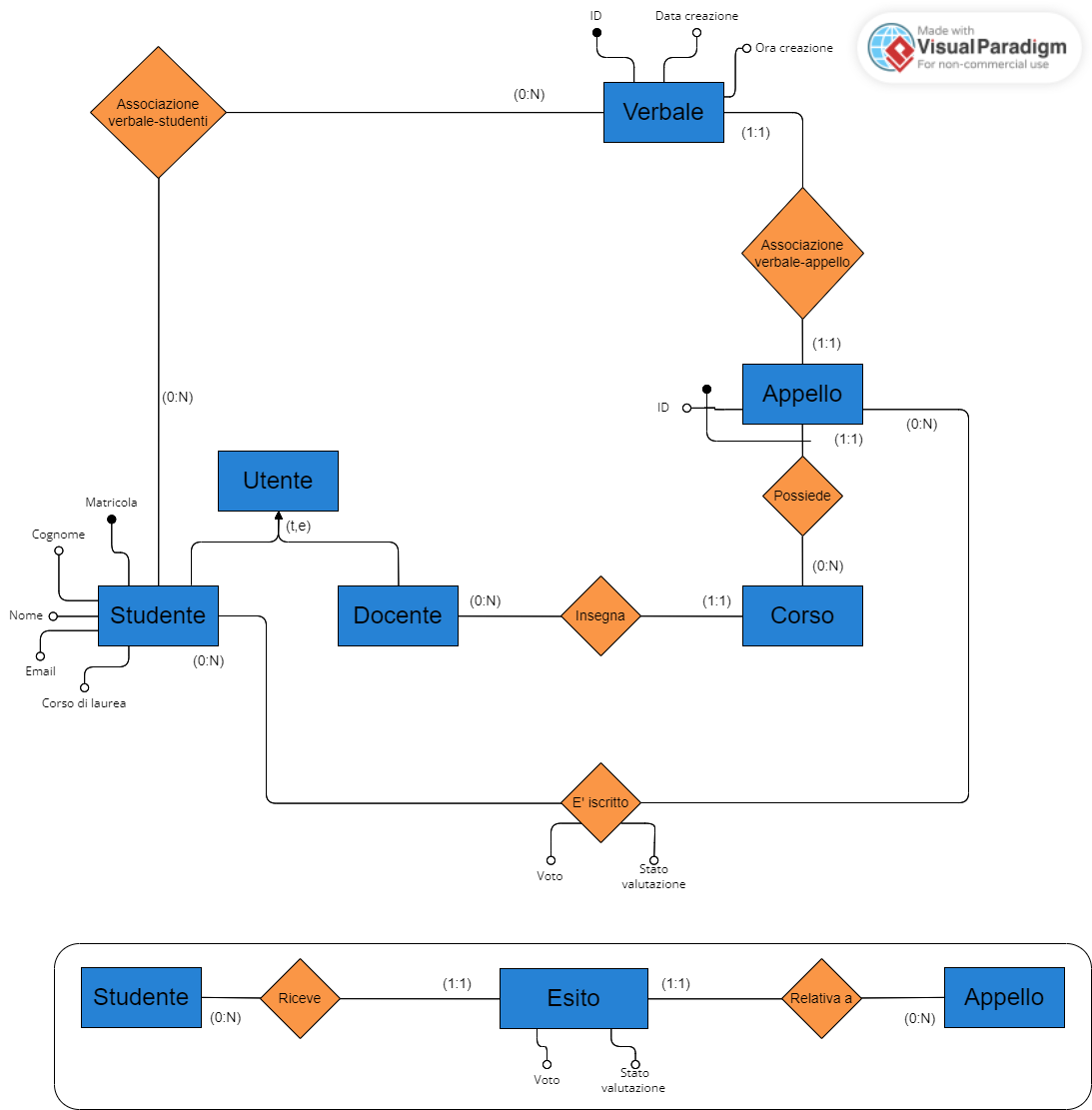
Un’applicazione permette di verbalizzare gli esiti degli esami di un appello. Il docente accede tramite login e seleziona nella HOME page un corso da una lista dei propri corsi ordinata in modo alfabetico decrescente e poi una data d’appello del corso scelto selezionata da un elenco ordinato per data 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: <vuoto>, assente, rimandato, riprovato, 18, 19, …, 30, 30 e lode. Nella tabella della pagina ISCRITTI 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 della pagina ISCRITTI 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 essere iscritto 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.

Entità, Attributi, Relazioni

**Database Design**



Alternativa alla relazione N:N “E’ iscritto”:



**Local DB Schema**

***Tabelle(ITA):***

Appelli (ID, ID\_Corso, Ora, Data)

Corsi (ID, Nome, Descrizione, ID\_Docente)

Corsi\_di\_laurea (ID, Nome, Descrizione)

Utenti (ID, Cognome, Nome, Email, Username, Password, Ruolo, ID\_CorsoDiLaurea\*)

Iscrizioni\_appelli (ID\_Studente, ID\_Appello, Voto, StatoValutazione)

Iscrizioni\_corsi (ID\_Studente, ID\_Corso)

Studenti\_Verbali (ID\_Studente, ID\_Verbale)

Verbali (ID, DataCreazione, OraCreazione, ID\_Appello)

**Commento**:

Per l’implementazione della tabella Appelli, ho valutato 2 possibili alternative:

* Mantenere la relazione di entità debole, e quindi realizzare la tabella in modo che la chiave primaria sia costituita dall’attributo OraData (Unione dei due attributi “Ora” e “Data”) e da ID\_Corso (chiave primaria della tabella Corso), questo permette di specificare che ad un corso siano associati più appelli a patto che siano in momenti diversi (OraData diversi). Il lato negativo di questa implementazione è che non è possibile specificare che più appelli relativi allo stesso corso possano essere tenuti alla stessa data e ora.
* (Scelta adottata) Trasformare la relazione di entità debole in una normale relazione 1:N introducendo un attributo “ID”, all’interno della tabella Appelli. In questo modo a differenza del caso precedente, posso anche andare a specificare la concomitanza di due appelli relativi allo stesso corso ed inoltre posso tenere separati gli attributi “Ora” e “Data” per maggiore comodità nelle query.

***Tabelle(ENG):***

Calls (ID, Time, Date,ID\_Course)

Courses (ID, Name, Description, ID\_Lecturer)

Degree\_courses (ID, Name, Description)

Users (ID, Surname, Name, Email, Username, Password, Role, ID\_DegreeCourse\*)

Registrations\_calls (ID\_Student, ID\_Call, Mark, EvaluationStatus)

Registrations\_courses (ID\_Student, ID\_Course)

Students\_verbals (ID\_Student, ID\_Verbal)

Verbals (ID, CreationDate, CreationTime, ID\_Call)

**Comment**:

For the Calls table I evaluated 2 possible alternatives:

* Maintain the weak relationship, and then create the table so that the primary key consists of the TimeDate attribute (Union of the “Time” and “Date” attributes) and ID\_Course (primary key of the Course table). This allows to specify that multiple calls can be associated with a course, provided that they are at different times (TimeDate must be different). The negative part of this implementation is that it’s not possible to specify that multiple calls related to the same course can be held on the same date and time.
* (Choice adopted) Transform the weak relationship into a normal 1:N relationship by introducing an “ID” attribute, in the Calls table. In this way, unlike the precedent case, I can specify the concurrence of two calls related to the same course and I can also keep the “Time” and “Date” attributes separated for more convenience in the Querys

***Chiavi esterne(ITA):***

corsi.ID\_Docente -> utenti.ID

studenti\_verbali.ID\_Studente -> utenti.ID

studenti\_verbali.ID\_Verbale -> verbali.ID

verbali.ID\_Appello -> appelli.ID

appelli.ID\_Corso -> corsi.ID

IscrizioniAppelli.ID\_Studente -> utenti.Matricola

IscrizioniAppelli.ID\_Appello -> appelli.ID

iscrizioniCorsi.ID\_Studente -> utenti.Matricola

iscrizioniCorsi.ID\_Appello -> appelli.ID

utenti.ID\_CorsoDiLaurea -> CorsiDiLaurea.ID

***Chiavi esterne(ENG):***

calls.ID\_Course -> courses.ID

courses.ID\_Lecturer -> lecturers.ID

Registrations\_calls.ID\_Student -> students.ID

Registrations\_calls.ID\_Call -> calls.ID

Registrations\_courses.ID\_Student -> students.ID

Registrations\_courses.ID\_Call -> calls.ID

users.ID\_DegreeCourse -> degree\_courses.ID

students\_verbals.ID\_Student -> students.ID

students\_verbals.ID\_Verbal -> verbals.ID

verbals.ID\_Call -> calls.ID

CREATE TABLE `calls` (

`ID` int NOT NULL AUTO\_INCREMENT,

`Date` date NOT NULL,

`Time` time NOT NULL,

`ID\_Course` int NOT NULL,

PRIMARY KEY (`ID`),

KEY `ID\_Course\_idx` (`ID\_Course`),

CONSTRAINT `Calls->Course` FOREIGN KEY (`ID\_Course`) REFERENCES `courses` (`ID`) ON UPDATE CASCADE

) ENGINE=InnoDB AUTO\_INCREMENT=8 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

CREATE TABLE `courses` (

`ID` int NOT NULL AUTO\_INCREMENT,

`Name` varchar(64) NOT NULL,

`Description` varchar(255) NOT NULL DEFAULT 'No description',

`ID\_Lecturer` int NOT NULL,

PRIMARY KEY (`ID`),

UNIQUE KEY `Name\_UNIQUE` (`Name`),

KEY `ID\_Docente\_idx` (`ID\_Lecturer`),

CONSTRAINT `Courses->Lecturers` FOREIGN KEY (`ID\_Lecturer`) REFERENCES `lecturers` (`ID`) ON

UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

CREATE TABLE `degree\_courses` (

`ID` int NOT NULL AUTO\_INCREMENT,

`Name` varchar(64) NOT NULL,

`Description` varchar(255) NOT NULL DEFAULT 'No description',

PRIMARY KEY (`ID`),

UNIQUE KEY `Nome\_UNIQUE` (`Name`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

CREATE TABLE `lecturers` (

`ID` int NOT NULL AUTO\_INCREMENT,

`Surname` varchar(64) NOT NULL,

`Name` varchar(64) NOT NULL,

`Email` varchar(64) NOT NULL DEFAULT 'Non indicata',

`Username` varchar(64) NOT NULL,

`Password` varchar(64) NOT NULL,

PRIMARY KEY (`ID`),

UNIQUE KEY `Email\_UNIQUE` (`Email`),

UNIQUE KEY `Username\_UNIQUE` (`Username`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

CREATE TABLE `registrations\_calls` (

`ID\_Student` int NOT NULL,

`ID\_Call` int NOT NULL,

`Mark` enum ('Assente', 'Rimandato', 'Riprovato', '18', '20', '21', '22', '23', '24', '25', '26', '27', '28',

'29','30','30 e lode') NOT NULL,

`EvaluationStatus` enum('Non inserito','Inserito','Pubblicato','Rifiutato','Verbalizzato') NOT NULL,

PRIMARY KEY (`ID\_Student`,`ID\_Call`),

KEY `ID\_Call\_idx` (`ID\_Call`),

CONSTRAINT `Registrations\_calls->Calls` FOREIGN KEY (`ID\_Call`) REFERENCES `calls` (`ID`) ON

UPDATE CASCADE,

CONSTRAINT `Registrations\_calls->Students` FOREIGN KEY (`ID\_Student`) REFERENCES `students`

(`ID`) ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

CREATE TABLE `registrations\_courses` (

`ID\_Student` int NOT NULL,

`ID\_Course` int NOT NULL,

PRIMARY KEY (`ID\_Student`,`ID\_Course`),

KEY `Registrations\_courses->Courses\_idx` (`ID\_Course`),

CONSTRAINT `Registrations\_courses->Courses` FOREIGN KEY (`ID\_Course`) REFERENCES `courses`

(`ID`) ON UPDATE CASCADE,

CONSTRAINT `Registrations\_courses->Students` FOREIGN KEY (`ID\_Student`) REFERENCES `students`

(`ID`) ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

CREATE TABLE `students` (

`ID` int NOT NULL AUTO\_INCREMENT,

`Surname` varchar(64) NOT NULL,

`Name` varchar(64) NOT NULL,

`Email` varchar(64) NOT NULL DEFAULT 'Non indicata',

`Username` varchar(64) NOT NULL,

`Password` varchar(64) NOT NULL,

`ID\_DegreeCourse` int NOT NULL,

PRIMARY KEY (`ID`),

UNIQUE KEY `Email\_UNIQUE` (`Email`),

UNIQUE KEY `Username\_UNIQUE` (`Username`),

KEY `Students->DegreeCourses\_idx` (`ID\_DegreeCourse`),

CONSTRAINT `Students->DegreeCourses` FOREIGN KEY (`ID\_DegreeCourse`) REFERENCES

`degree\_courses` (`ID`) ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

CREATE TABLE `students\_verbals` (

`ID\_Student` int NOT NULL,

`ID\_Verbal` int NOT NULL,

PRIMARY KEY (`ID\_Student`,`ID\_Verbal`),

KEY `students\_verbals->Verbals\_idx` (`ID\_Verbal`),

CONSTRAINT `students\_verbals->Students` FOREIGN KEY (`ID\_Student`) REFERENCES `students`

(`ID`) ON UPDATE CASCADE,

CONSTRAINT `students\_verbals->Verbals` FOREIGN KEY (`ID\_Verbal`) REFERENCES `verbals` (`ID`)

ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

CREATE TABLE `verbals` (

`ID` int NOT NULL AUTO\_INCREMENT,

`CreationDate` date NOT NULL,

`CreationTime` time NOT NULL,

`ID\_Call` int NOT NULL,

PRIMARY KEY (`ID`),

KEY `Verbals->Calls\_idx` (`ID\_Call`),

CONSTRAINT `Verbals->Calls` FOREIGN KEY (`ID\_Call`) REFERENCES `calls` (`ID`) ON UPDATE

CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

**Analisi requisiti applicazione**

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: <vuoto>, assente, rimandato, riprovato, 18, 19, …, 30, 30 e lode. Nella tabella della pagina ISCRITTI 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 della pagina ISCRITTI 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 essere iscritto 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.

**Pagine (Viste), Componenti viste, Eventi, Azioni**

**Componenti**

* Model Objects (Beans)
  + CallEvaluation
  + Course
  + DegreeCourse
  + GraduationCall
  + User
  + Verbal
* Data Access Objects (Classes)
  + CallEvaluationDAO
    - FindAllEvaluationByStudentId(int student\_id) : List<CallEvaluation>
    - FindAllEvaluationByCallId(int call\_id) : List<CallEvaluation>
    - FindEvaluationByCallAndStudentId(int call\_id, int student\_id) : CallEvaluation
    - UpdateEvaluationStateByStudentAndCallId(int student\_id, int call\_id, String evaluationState) : int
    - VerbalizeAllMarksByCallId(Date verbalDate, Time verbalTime, int call\_id) : int
    - PublishAllMarksByCallId(int call\_id) : int
    - UpdateMarkByStudentAndCallId(int student\_id, int call\_id, String newMark) : int
    - CheckIfAnyMarkIsVerbalizable() : void
    - CheckIfMarkFormatIsCorrect(String mark) : void
    - GetNumberOfVerbalizableMarks() : int
    - CheckIfStudentMarkIsUpdatable(int student\_id, int call\_id) : void
  + CourseDAO
    - findAllCoursesByLecturer(int lecturer\_id) : List<Course>
    - FindAllCourseByStudentId(int student\_id) : List<Course>
    - findCourseById(int course\_id) : Course
    - CheckIfCourseIsTaughtByLecturer(int course\_id, int lecturer\_id) : void
  + DegreeCourseDAO
    - findAllDegreeCourses() : List<DegreeCourse>
    - findDegreeCourseById(int degreeCourse\_id) : DegreeCourse
  + GraduationCallDAO
    - findAllDegreeCallByCourseId(int course\_id) : List<GraduationCall>
    - FindAllDegreeCallWhichStudentSubscribedToByCourseId(int student\_id, int course\_id) : List<GraduationCall>
    - findAllDegreeCallByDate(Date date) : List<GraduationCall>
    - createGraduationCall(Date date, Time time, int id\_course) : int
    - GetGraduationCallById(int call\_id) : GraduationCall
    - CheckIfCourseOfCallIsTaughtByLecturer(int call\_id, int lecturer\_id) : void
  + LecturerDAO
    - InsertLecturer(String surname, String name, String email, String username, String password) : int
    - findLecturerById(int lecturer\_id) : User
  + StudentDAO
    - findAllStudentsByDegreeCourse(int degreeCourse\_id) : List<User>
    - insertStudent(String surname, String name, String email, String username, String password, int id\_degreeCourse) : int
    - findStudentsInVerbal(int verbal\_id) : List<User>
    - findAllRegistrationsToTheCall(int call\_id) : List<User>
    - FindAllRegistrationsToTheCall(int call\_id, String orderBy, String orderType) : List<User>
    - findStudentById(int student\_id) : User
    - FindAllRegistrationsAndEvaluationToCall(int call\_id) : Map<User, CallEvaluation>
    - FindAllRegistrationsAndEvaluationToCallOrdered(int call\_id, String orderBy, String orderType) : Map<User, CallEvaluation>
    - CheckIfStudentIsSubscribedToCourse(int student\_id, int course\_id) : void
    - CheckIfStudentIsSubscribedToCall(int student\_id, int call\_id) : void
    - FindAllStudentsInVerbalById(int verbal\_id) : List<User>
  + UserDAO
    - CheckCredentials(String username, String password) : User
  + VerbalDAO
    - findAllVerbalsByCall(int call\_id) : List<Verbal>
    - CreateVerbal(Date creationDate, Time creationTime, int call\_id) : int
    - SaveStudentWithinVerbal(int call\_id, List<User> students) : int
    - GetVerbalById(int verbal\_id) : Verbal
    - GetVerbalByCallId(int call\_id) : Verbal
    - GetVerbalByCallIdDateTime(Date date, Time time, int call\_id) : Verbal
* Controllers (servlets)
  + CheckLogin
  + GetSubscriptionToCall
  + GoToHomeLecturer
  + GoToHomeStudent
  + GoToMarkManagement
  + GoToOutcome
  + GoToVerbalRecap
  + Logout
  + RefuseMark
  + UpdateStudentMark
  + VerbalizeStudentsMarks
* Views (Templates)
  + index
  + HomeLecturer
  + HomeStudent
  + Subscribers
  + MarkManagement
  + Outcome
  + Verbal