Tecnologie Informatiche per il Web - AA. 2020-21

Stefano Bagarin, Gregorio Galletti, Andrea Donati

June 28, 2021

Scopo del documento

Questo documento ha come obiettivo la presentazione delle specifiche tecniche implementative adottate per la realizzazione del progetto finale relativo al corso di Tecnologie Informatiche per il Web.

Il progetto consiste nello sviluppo di un'applicazione web in due differenti versioni:

- Versione HTML pura: si tratta di un'architettura thin client in cui il server gestisce la parte di presentazione tramite l'utilizzo di Thymeleaf.
- Versione Javascript: architettura thick client in cui la parte di presentazione e parte della logica dell'applicazione vengono gestite lato client.

Specifiche del progetto - Verbalizzazione degli esami - 1

Nelle pagine seguenti viene riportata la consegna dataci per implementare il progetto.

Nel testo vengono messe in evidenza le entità con gli attributi e le relazioni esistenti tra esse che risultano direttamente dalla consegna, ciascuna con un colore diverso:

- `Rosso:` le entità che si ritrovano nel database sotto forma di tabelle.
- `Verde:` gli attributi fondamentali delle entità.
- `Blu:` le relazioni esistenti tra le varie entità

Versione HTML pura

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. Ad ogni riga corrisponde un bottone "MODIFICA".

Specifiche del progetto - Verbalizzazione degli esami - 3

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.

Specifiche del progetto - Verbalizzazione degli esami - 4

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.

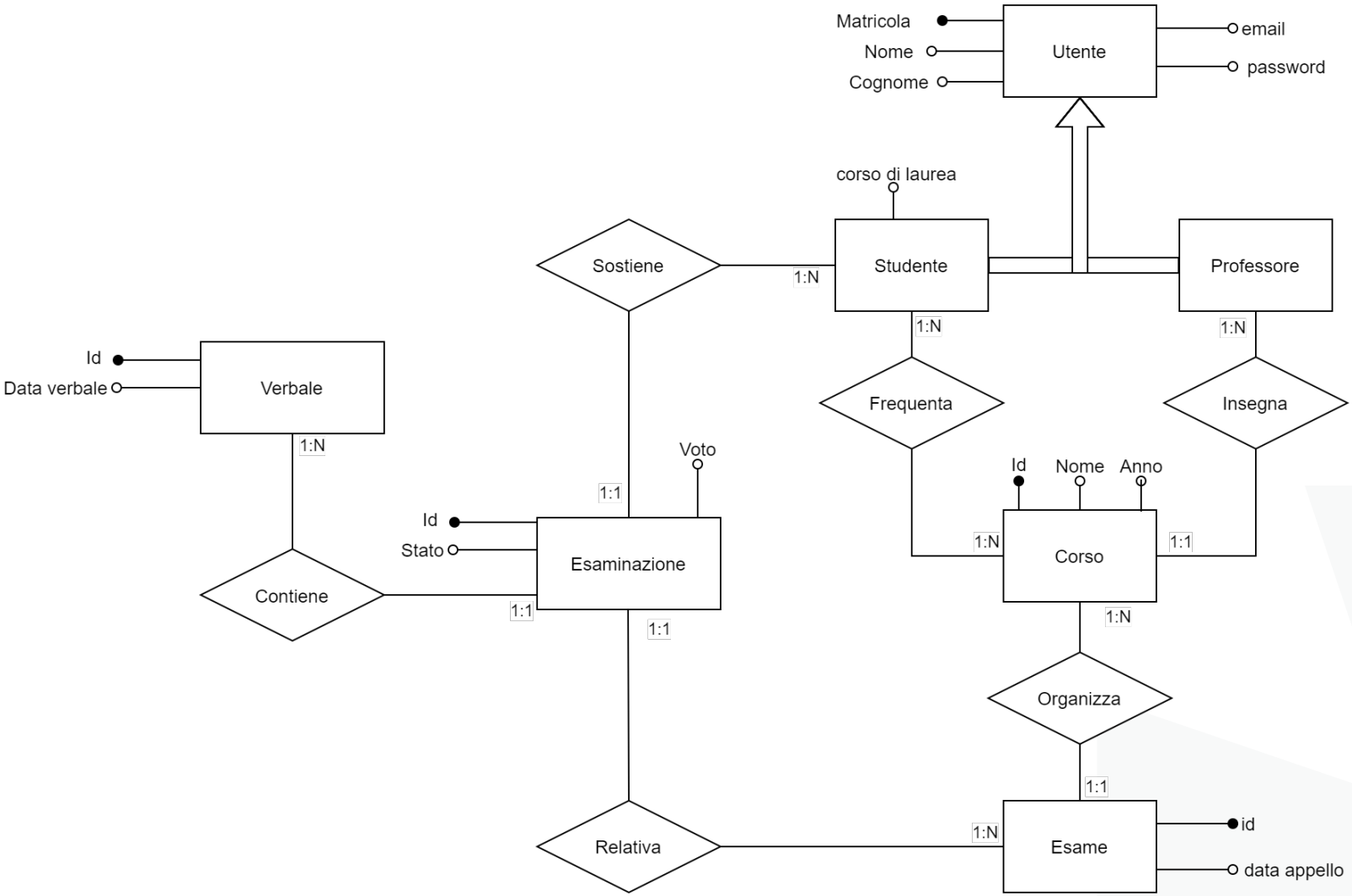
Il rifiuto implica la verbalizzazione di "rimandato" come voto. Un verbale ha un codice generato dal sistema, una data e ora di creazione ed è associato all'appello del corso a cui si riferisce e agli studenti (con nome, cognome, matricola e voto) che passano allo stato "verbalizzato". A seguito della pressione del bottone VERBALIZZA compare una pagina VERBALE che mostra i dati completi del verbale creato.

Versione con JavaScript

Questa versione del progetto prevede le seguenti modifiche alle specifiche appena illustrate:

- Dopo il login dell'utente, l'intera applicazione è realizzata con un'unica pagina per il docente e un'unica pagina per lo studente.
- Ogni interazione dell'utente è gestita senza ricaricare completamente la pagina, ma produce l'invocazione asincrona del server e l'eventuale modifica del contenuto da aggiornare a seguito dell'evento.
- La funzione di riordino della tabella degli iscritti è realizzata a lato client.
- Alla tabella degli iscritti è associato un bottone INSERIMENTO MULTIPLO che provoca la comparsa di una pagina modale con tutte e sole le righe nello stato "non inserito" associate a un campo di input. Il docente può inserire un voto per un insieme delle righe e premere un bottone INVIA che comporta l'invio al server dei voti, il cambio di stato delle righe coinvolte, la chiusura della finestra modale e l'aggiornamento della tabella degli iscritti.

Design del database



Schema del database - Utente

```
CREATE TABLE `sys`.`utente` (
  `matricola` INT NOT NULL AUTO_INCREMENT,
  `nome` VARCHAR(45) NOT NULL,
  `cognome` VARCHAR(45) NOT NULL,
  `email` VARCHAR(45) NOT NULL,
  `password` VARCHAR(45) NOT NULL,
  `role` ENUM('student', 'teacher') NOT NULL,
  `cdl` VARCHAR(45) NULL DEFAULT NULL,
  PRIMARY KEY (`matricola`),
  UNIQUE INDEX `email_UNIQUE` (`email` ASC) VISIBLE);
```

Schema del database - Corso

```
CREATE TABLE `sys`.`corso` (
  `id` INT NOT NULL,
  `matricolaProfessore` INT NOT NULL,
  `nomeCorso` VARCHAR(45) NOT NULL,
  `annoCorso` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_matricolaProfessore_idx` (`matricolaProfessore` ASC) VISIBLE,
  CONSTRAINT `fk_matricolaProfessore`
    FOREIGN KEY (`matricolaProfessore`)
    REFERENCES `sys`.`utente` (`matricola`)
    ON DELETE CASCADE
    ON UPDATE CASCADE);
```

Schema del database - Esame

```
CREATE TABLE `sys`.`esame` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `idCorso` INT NOT NULL,
  `dataAppello` DATE NOT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_corso_idx` (`idCorso` ASC) VISIBLE,
  CONSTRAINT `fk_corso`
    FOREIGN KEY (`idCorso`)
    REFERENCES `sys`.`corso` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE);
```

Schema del database - Frequentazione

```
CREATE TABLE `sys`.`frequentazione` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `matricolaStudente` INT NOT NULL,
  `idCorso` INT NOT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_idCorso_idx` (`idCorso` ASC) VISIBLE,
  INDEX `fk_matricolaStudente_idx` (`matricolaStudente` ASC) VISIBLE,
  CONSTRAINT `fk_idCorso`
    FOREIGN KEY (`idCorso`)
    REFERENCES `sys`.`corso` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_matricolaStudente`
    FOREIGN KEY (`matricolaStudente`)
    REFERENCES `sys`.`utente` (`matricola`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION);
```

Schema del database - Esaminazione

```
CREATE TABLE `sys`.`esaminazione` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `idStudente` INT NOT NULL,
  `idEsame` INT NOT NULL,
  `idVerbale` INT NULL DEFAULT NULL,
  `voto` ENUM(' ', 'assente', 'rimandato', 'riprovato', '18', '19', '20', '21', '22', '23', '24', '25', '26', '27', '28', '29', '30', '30 e lode') NOT NULL DEFAULT ' ',
  `stato` ENUM('non inserito', 'inserito', 'pubblicato', 'rifiutato', 'verbalizzato') NOT NULL DEFAULT 'non inserito',
  PRIMARY KEY (`id`),
  INDEX `fk_idEsame_idx` (`idEsame` ASC) VISIBLE,
  INDEX `fk_idStudente_idx` (`idStudente` ASC) VISIBLE,
  INDEX `fk_idVerbale_idx` (`idVerbale` ASC) VISIBLE,
  CONSTRAINT `fk_idEsame`
    FOREIGN KEY (`idEsame`)
    REFERENCES `sys`.`esame` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `fk_idStudente`
    FOREIGN KEY (`idStudente`)
    REFERENCES `sys`.`utente` (`matricola`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `fk_idVerbale`
    FOREIGN KEY (`idVerbale`)
    REFERENCES `sys`.`verbale` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE);
```

Schema del database - Verbale

```
CREATE TABLE `sys`.`verbale` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `dataVerbale` DATETIME NOT NULL,
  PRIMARY KEY (`id`));
```

Analisi dei requisiti dell'apllicazione - 1

Nelle pagine seguenti viene riportata l'analisi dei requisiti dell'applicazione eseguita evidenziando le views, i componenti delle views, gli eventi e le azioni, ciascuna con un colore diverso:

- `Rosso:` views.
- `Verde:` componenti delle view.
- `Blu:` eventi.
- `Marrone:` azioni.

Versione HTML pura

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. Ad ogni riga corrisponde un bottone "MODIFICA".

Analisi dei requisiti dell'apllicazione - 3

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.

Analisi dei requisiti dell'apllicazione - 4

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.

Versione con JavaScript

Questa versione del progetto prevede le seguenti modifiche alle specifiche appena illustrate:

- Dopo il login dell'utente, l'intera applicazione è realizzata con un'unica pagina per il docente e un'unica pagina per lo studente.
- Ogni interazione dell'utente è gestita senza ricaricare completamente la pagina, ma produce l'invocazione asincrona del server e l'eventuale modifica del contenuto da aggiornare a seguito dell'evento.
- La funzione di riordino della tabella degli iscritti è realizzata a lato client.
- Alla tabella degli iscritti è associato un bottone INSERIMENTO MULTIPLO che provoca la comparsa di una pagina modale con tutte e sole le righe nello stato "non inserito" associate a un campo di input. Il docente può inserire un voto per un insieme delle righe e premere un bottone INVIA che comporta l'invio al server dei voti, il cambio di stato delle righe coinvolte, la chiusura della finestra modale e l'aggiornamento della tabella degli iscritti.

Completamento e aggiunta delle specifiche - pure HTML - 1

- L'applicazione è realizzata con un'unica pagina login comune a docenti e studenti.
- Un utente si può autenticare attraverso l'inserimento del proprio codice matricola e password. Se questi corrispondono ad utente esistente, esso viene reindirizzato alla propria HOME page, altrimenti viene mostrato un messaggio di errore.
- In qualsiasi momento dopo l'autenticazione sono sempre disponibili sulla barra di navigazione un bottone di logout che reindirizza alla pagina di login e un bottone per ritornare alla HOME page.
- In qualsiasi momento dopo l'autenticazione sono sempre visibili in una barra laterale foto, nome, cognome, email e matricola dell'utente,
- Lo stato di autenticazione di un utente è tenuto tramite sessione, nel momento in cui questa scade, alla prossima interazione con il server l'utente verrà reindirizzato alla pagina di login.
- Dopo aver selezionato un corso dalla lista dei propri corsi, viene visualizzata una lista degli anni in cui è stato presente quel corso, ciascuno con gli appelli relativi.
- In tutte le pagine dopo alla home page è presente un bottone che consente di ritornare alla pagina precedente.
- L'invio della form di modifica del voto di uno studente da parte di un docente causa una reindirizzazione alla pagina ISCRITTI relativa all'appello.

Completamento e aggiunta delle specifiche - pure HTML - 2

- Se un docente non è il responsabile di nessun corso, nella HOME page viene visualizzato il messaggio "Non sei il docente di nessun corso al momento".
- Se uno studente non è iscritto a nessun corso, nella HOME page viene visualizzato il messaggio "Non sei iscritto a nessun corso al momento".
- Se uno studente non è iscritto a nessun appello di uno specifico anno di un corso, viene visualizzato il messaggio "Non sei iscritto/a a nessun appello di questo corso." sotto di esso.
- Se non sono ancora presenti appelli per un corso tenuto da un docente, viene visualizzato il messaggio "Non sono presenti appelli per questo corso.".
- Se nessuno studente è iscritto ad un appello, la pagina ISCRITTI relativa a quell'appello mostrerà il messaggio "Nessuno studente iscritto all'appello.".

Completamento e aggiunta delle specifiche - RIA

In aggiunta a quelle già elencate per la versione pure HTML si ha:

- Nella pagina modale di inserimento multiplo è possibile selezionare più righe e inserire contemporaneamente lo stesso voto per ciascuna di esse.
- Nella pagina ISCRITTI, la pressione dei bottoni PUBBLICA e VERBALIZZA causa la comparsa di un messaggio di successo/fallimento in basso alla pagina.

Componenti - pure HTML

Controllers

- Login
- Logout
- Home
- ElencoEsami
- GetResults
- InserisciVoti
- ModificaVoto
- PubblicaVoti
- RifiutaVoti
- VerbalizzaVoti
- RootServlet

Daos

- CorsoDAO
- EsameDAO
- EsaminazioneDAO
- UserDAO
- VerbaleDAO

Beans

- Corso
- Esame
- Esaminazione
- User
- Verbale

Filtri

- AuthorizationChecker
- LoginChecker
- StudentChecker

Templates

- index.html
- Home.html
- ElencoEsami.hmtl
- RisultatiEsame.html
- ModificaVoto.html
- Verbale.html

Componenti lato server - RIA

Controllers

- Login
- Logout
- Home
- ElencoEsami
- GetResults
- GetCorsi
- GetInfoUtente
- InserisciVoti
- InserimentoMultiplo
- PubblicaVoti
- RifiutaVoti
- VerbalizzaVoti
- RootServlet

Daos

- CorsoDAO
- EsameDAO
- EsaminazioneDAO
- UserDAO
- VerbaleDAO

Beans

- Corso
- Esame
- Esaminazione
- User
- Verbale

Filtri

- AuthorizationChecker
- LoginChecker
- StudentChecker

Templates

- index.html
- Home.html

Componenti lato client - RIA - 1

Pagina di login

- makePost()
- showResults()

Pagina principale

- showInfo(request)
- showCorsi(request)
- getEsami()
- showEsami(request)
- getRisultati()
- showRisultati(request)
- resetTable()
- inserimentoMultiplo()
- modificaVoti(matricola,voto)
- inserisciVoti()
- rifiutaVoto()
- pubblicaVoti()
- verbalizzaVoti()
- showVerbale(request)

Gestione tabella iscritti

- ordinaTabella(n)
- comparaVoti(voto1,voto2,ordine)
- compareRows(x,y,ordine,n)

Gestione inserimento multiplo

- selezionaTutto()
- selezionaRiga(id)
- areAllSelected()
- handleSelection(selection)

Componenti lato client - RIA - 2

Funzioni di utilità per creare elementi HTML dinamicamente

- createDiv(classes, id, role)
- createLabelledDiv(classes, id, role, labelledby)
- createText(tag, classes, id)
- createAccordionText(tag)
- createBtn(tag, classes, text)
- createAccordionBtn(text, i)
- createIcon(classes)
- createOptions(empty)
- createSelect(id, empty)
- createCell(text)
- createHeaderCell(text)
- createCheckbox(id)
- createHidden()
- createForm()
- createTabella(studente, voto)
- createTabellaVerbale()

Design applicativo (IFML) - pure HTML

Design applicativo (IFML) - RIA

Eventi e azioni - 1

Lato Client		Lato Server		Utente
Evento	**Azione**	**Evento**	**Azione**	
Pagina Login -> login	-	POST(matricola, password)	Controllo validità credenziali e ruolo utente	Docente, Studente
Pagina Home -> carica	Mostra dati e corsi dell'utente	GET()	Recupera dati e informazioni sui corsi dell'utente	Docente, Studente
Pagina Home -> selezione corso	Mostra lista degli anni in cui è stato presente il corso selezionato	GET(nome corso)	Recupera lista di anni e appelli del corso	Docente, Studente
Lista corso-anni -> selezione anno	Mostra lista degli appelli relativi al corso dell'anno selezionato	-	-	Docente, Studente
Lista corso-anni -> freccia indietro	Mostra dati e corsi dell'utente	GET()	Recupera dati e informazioni sui corsi dell'utente	Docente, Studente
Lista appelli -> selezione appello	Mostra una tabella con i voti degli studenti iscritti all'appello	GET(id appello)	Recupera la lista degli studenti iscritti all'appello e i voti conseguiti	Docente
Tabella iscritti -> bottone "Modifica"	Mostra un form che consente la modifica del voto di uno studente	-	-	Docente
Form modifica voto -> bottone "Applica"	Mostra la tabella con i voti degli studenti iscritti all'appello	POST(matricola studente, voto, id appello)	Inserisce il voto dello studente per l'appello corrente	Docente
Form modifica voto -> freccia indietro	Mostra una tabella con i voti degli studenti iscritti all'appello	GET(id appello)	Recupera la lista degli studenti iscritti all'appello e i voti conseguiti	Docente
Tabella iscritti -> bottone "Inserimento multiplo"	Mostra un modal per l'inserimento di più voti contemporaneamente	-	-	Docente

Eventi e azioni - 2

Lato Client		Lato Server		Utente
Evento	**Azione**	**Evento**	**Azione**	
Modal inserimeno multiplo -> bottone “Inserisci”	Chiude il modal	POST(id appello, coppie di studente e voto)	Inserisci i voti degli studenti selezionati per l’appello corrente	Docente
Tabella iscritti -> bottone “Pubblica”	Mostra la tabella con i voti degli studenti iscritti all’appello	GET(id appello) GET(id appello)	Cambia lo stato delle valutazioni degli studenti con “inserito” in “pubblicato” e recupera i voti	Docente
Tabella iscritti -> bottone “Verbalizza”	Mostra i dati del verbale creato	GET(id appello)	Cambia lo stato delle valutazioni degli studenti con “pubblicato” in “verbalizzato” e recupera i dati degli studenti	Docente
Dati verbale -> freccia indietro	Mostra la tabella con i voti degli studenti iscritti all’appello	GET(id appello)	Recupera la lista degli studenti iscritti all’appello e i voti conseguiti	Docente
Tabella iscritti -> freccia inietro	Mostra lista degli anni in cui è stato presente il corso selezionato	GET(nome corso)	Recupera lista di anni e appelli del corso	Docente
Lista appelli -> selezione appello	Mostra l’esito conseguito nell’appello	GET(id appello)	Recupera il voto preso dallo studente nell’appello selezionato	Studente
Visualizzazione esito appello -> bottone “Rifiuta”	Mostra l’esito conseguito nell’appello	GET(id appello)	Cambia lo stato della valutazione dello studente in “rifiutato”	Studente
Visualizzazione esito appello -> freccia indietro	Mostra lista degli anni in cui è stato presente il corso selezionato	GET(nome corso)	Recupera lista di anni e appelli del corso	Studente
Logout	-	GET()	Finisce la sessione e reindirizza alla pagina di login	Docente, Studente
Bottone Home	Mostra dati e corsi dell’utente	GET()	Recupera dati e informazioni sui corsi dell’utente	Docente, Studente

Eventi e controllori - 1

Lato Client		Lato Server		Utente
Evento	**Controllore**	**Evento**	**Controllore (servlet)**	
Pagina Login -> login	makePost()	POST(matricola, password)	Login	Docente, Studente
Pagina Home -> carica	showInfo() showCorsi()	GET()	GetInfoUtente GetCorsi	Docente, Studente
Pagina Home -> selezione corso	getEsami() showEsami()	GET(nome corso)	ElencoEsami	Docente, Studente
Lista corso-anni -> selezione anno	-	-	-	Docente, Studente
Lista corso-anni -> freccia indietro	showCorsi()	GET()	GetCorsi	Docente, Studente
Lista appelli -> selezione appello	getRisultati() showRisultati()	GET(id appello)	GetResults	Docente
Tabella iscritti -> bottone “Modifica”	modificaVoti()	-	-	Docente
Form modifica voto -> bottone “Applica”	getRisultati() showRisultati()	POST(matricola studente, voto, id appello)	InserisciVoti	Docente
Form modifica voto -> freccia indietro	getRisultati() showRisultati()	GET(id appello)	GetResults	Docente
Tabella iscritti -> bottone “Inserimento multiplo”	inserimentoMultiplo()	-	-	Docente

Eventi e controllori - 2

Lato Client		Lato Server		Utente
Evento	**Controllore**	**Evento**	**Controllore (servlet)**	
Modal inserimeno multiplo -> bottone “Inserisci”	showRisultati()	POST(id appello, coppie di studente e voto)	InserimentoMultiplo	Docente
Tabella iscritti -> bottone “Pubblica”	pubblicaVoti() showRisultati()	GET(id appello) GET(id appello)	PubblicaVoti	Docente
Tabella iscritti -> bottone “Verbalizza”	verbalizzaVoti() showVerbale()	GET(id appello)	VerbalizzaVoti	Docente
Dati verbale -> freccia indietro	getRisultati() showRisultati()	GET(id appello)	GetResults	Docente
Tabella iscritti -> freccia inietro	getEsami() showEsami()	GET(nome corso)	ElencoEsami	Docente
Lista appelli -> selezione appello	getRisultati() showRisultati()	GET(id appello)	GetResults	Studente
Visualizzazione esito appello -> bottone “Rifiuta”	rifiutaVoto() showRisultati()	GET(id appello)	RifiutaVoti	Studente
Visualizzazione esito appello -> freccia indietro	getEsami() showEsami()	GET(nome corso)	ElencoEsami	Studente
Logout	-	GET()	Logout	Docente, Studente
Bottone Home	-	GET()	GetCorsi	Docente, Studente

Sequence diagrams