

Sistema di Voto e Scrutinio Elettronico

Relazione

Lucia Anna Mellini
matricola 943134

Anno Accademico 2021/22

Indice

1	Descrizione del sistema	3
1.1	Glossario	3
2	Specifica dei requisiti	5
2.1	Requisiti funzionali	5
2.1.1	Requisiti degli utenti	5
2.1.2	Requisiti del sistema	8
2.2	Requisiti non funzionali	11
2.2.1	Requisiti di qualità del software	11
2.2.2	Requisiti di sicurezza	12
2.2.3	Requisiti degli utenti propedeutici all'utilizzo del sistema	13
2.2.4	Requisiti per la fase di testing	14
3	Progettazione	15
3.1	Diagramma dei casi d'uso	15
3.1.1	Descrizione degli scenari dei casi d'uso	15
3.2	Diagramma delle classi	19
3.2.1	Diagramma delle classi di <i>model</i>	19
3.2.2	Diagramma delle classi di <i>database</i>	19
3.2.3	Diagramma delle classi di <i>controllers-model</i>	19
3.3	Diagrammi delle attività	21
3.4	Diagrammi di sequenza	21
3.5	Macchine di stato	24
3.6	Diagramma delle componenti	24
3.7	Diagramma di deployment	24
4	Implementazione	26
4.1	Diagramma delle classi	26
4.1.1	Diagramma delle classi di <i>model</i>	26
4.1.2	Diagramma delle classi di <i>database</i>	32
4.2	Vincoli	34
4.2.1	Vincoli OCL	34
4.2.2	Vincoli JML	39
4.3	Diagrammi di sequenza	41
4.4	Macchine di stato	41
4.5	Descrizione Design Pattern utilizzati	42
4.5.1	DAO	42
4.5.2	MVC	42
4.5.3	Singleton	44
4.5.4	Observer	44
4.6	Discussione scelte high cohesion - low coupling	45
4.7	Gestione dati persistenti	46
4.8	Descrizione interfaccia grafica	47
4.9	Testing	51
4.9.1	Classi di <i>model</i>	51
4.9.2	Interfaccia grafica	60
4.9.3	Database	60

5	Considerazioni finali	61
5.1	Discrepanze progettazione - implementazione	61
5.2	Sviluppi futuri	61
5.3	Note per l'installazione e l'utilizzo	62
5.3.1	Contenuto del database	62

Capitolo 1

Descrizione del sistema

1.1 Glossario

Al fine di una comprensione non ambigua della relazione in oggetto, segue un glossario teso ad esplicitare il significato sotteso ai termini indicati.

Attori coinvolti

Utente chiunque fa utilizzo del sistema.

Elettore utente che ha l'obiettivo di svolgere il procedimento di voto.

Componente dell'Amministrazione Elettorale (CAE) utente che a seconda dei propri permessi ha l'obiettivo di svolgere l'inserimento del contenuto delle schede tradizionali, il procedimento di configurazione e/o il procedimento di scrutinio.

Gestore di sistema la figura che può modificare il sistema per quanto concerne il suo funzionamento.

Principali casi d'uso

Autenticazione fase in cui il sistema assegna all'utente la propria identità digitale e verifica se l'utente è un elettore, CAE o entrambi. Nel caso in cui l'elettore abbia votato in sede questa fase è eseguita in presenza dal CAE.

Procedimento di voto prevede la scelta della scheda elettorale da compilare, la compilazione della scheda scelta in base alla modalità di voto, la conferma della preferenza espressa e l'invio della notifica di corretta registrazione all'elettore.

Procedimento di configurazione prevede le operazioni di creazione di nuove sessioni e delle sue componenti.

Procedimento di scrutinio ha inizio al termine delle consultazioni; prevede il conteggio dei voti in base alla modalità del calcolo del vincitore e la successiva comunicazione dei risultati agli OPRU.

Terminologia votazioni

Consultazione include i procedimenti di voto, l'inserimento dei voti tradizionali e il procedimento di scrutinio. Può essere politica, amministrativa, del Parlamento europeo, oppure un referendum. In una sessione possono esserci molteplici consultazioni congiunte.

Periodo di voto intervallo temporale impostato da un CAE autorizzato, entro il quale è possibile compiere i procedimenti di voto e il CAE può inserire il contenuto delle schede tradizionali per una certa consultazione.

Sessione di voto aggrega molteplici consultazioni in corso durante il medesimo periodo di voto. Le sue componenti sono tutte le consultazioni che la compongono, e le relative componenti, quali eventuali candidati, partiti e quesiti.

Modalità di voto

Voto ordinale si richiede l'ordinamento di voci, che possono essere candidati o partiti.

Voto categorico si richiede l'inserimento di una preferenza per un solo candidato o partito.

Voto categorico con preferenze si richiede l'inserimento di una preferenza per un partito, e un'eventuale preferenza di un candidato di tale partito.

Referendum si richiede una scelta tra le opzioni *Sì* e *No* a un determinato quesito.

Modalità di calcolo del vincitore

Maggioranza il vincitore è il candidato o il partito che ha ottenuto il maggior numero di voti.

Maggioranza assoluta il vincitore è il candidato che ha ottenuto la maggioranza assoluta dei voti, cioè il $50\% + 1$ dei voti espressi.

Livello di quorum percentuale minima degli aventi diritti al voto che devono partecipare alla consultazione per poter procedere al conteggio dei voti durante il procedimento di scrutinio

Sigle

OPRU Organi preposti dalla legge per la Proclamazione dei Risultati Ufficiali.

SPID Sistema Pubblico di Identità Digitale.

CIE Carta d'Identità Elettronica italiana.

AgID Agenzia per l'Italia Digitale.

Capitolo 2

Specifica dei requisiti

Per ciascun requisito sono previsti un identificatore, utilizzato nelle sezioni a seguire, una descrizione e la motivazione dell'esistenza del requisito per il sistema in esame. Ove ritenuto necessario i requisiti sono stati sviluppati in sottopunti, per i quali la motivazione, se non esplicitata, è da ricercare nel requisito genitore. I punti sono dipendenti dai relativi sottopunti; le dipendenze si riferiscono ai punti indicati e ai relativi sottopunti.

Ove possibile i requisiti sono stati raggruppati secondo i casi d'uso del diagramma [3.1](#).

2.1 Requisiti funzionali

2.1.1 Requisiti degli utenti

Autenticazione

FU1

Descrizione L'utente deve eseguire l'autenticazione.

Motivazione Verificare la presenza dell'identità digitale nel database nazionale.

FU1.1

Descrizione L'utente deve scegliere il ruolo che vuole assumere nella sessione corrente.

Motivazione Mostrare le operazioni adatte al ruolo dell'utente.

Dipendenze *FS1*

Procedimento di voto

FU2

Descrizione L'utente deve poter compilare le schede elettorali per esprimere i voti.

Motivazione Obiettivo del sistema.

FU2.1

Descrizione Il CAE deve poter inserire il contenuto delle schede tradizionali nel sistema.

Motivazione Permettere di considerare nel procedimento di scrutinio anche il contenuto delle schede compilate manualmente.

Dipendenze *FS2*

FU2.1.1

Descrizione Il CAE deve confermare il contenuto della scheda inserita prima di terminare il procedimento di voto.

Motivazione Assicurare che la scelta registrata sia corrispondente alle intenzioni del CAE.

FU2.1.2

Descrizione Il CAE deve poter inserire gli elettori che hanno espresso la loro preferenza per una data consultazione.

Motivazione Assicurare un corretto scrutinio, in particolare un corretto conteggio degli elettori partecipanti per la valutazione del raggiungimento del quorum.

FU2.2

Descrizione L'elettore deve poter effettuare il procedimento di voto per le consultazioni.

Motivazione Obiettivo del sistema.

Dipendenze *FS3*

FU2.2.1

Descrizione Prima dell'inizio del procedimento di voto il sistema deve dichiarare il tempo (limitato) che sarà necessario per l'espressione e la conferma del voto.

Motivazione Rendere il sistema accessibile all'elettore.

FU2.2.2

Descrizione L'elettore deve confermare la propria preferenza dopo averla espressa.

Motivazione Assicurare che la scelta registrata sia corrispondente alla volontà dell'elettore.

FU2.2.3

Descrizione All'elettore deve arrivare una notifica di registrazione avvenuta del voto.

Motivazione Garantire trasparenza del sistema all'elettore.

Dipendenze *FS3.2*

FU2.3

Descrizione Nel caso di consultazioni congiunte l'elettore deve scegliere per quale effettuare il procedimento di voto.

Motivazione Permettere agli elettori di esercitare il diritto di voto per tutte le consultazioni attive durante il periodo di voto corrente.

FU2.4

Descrizione L'elettore deve indicare un voto valido oppure esercitare la facoltà di astenersi dalla scelta (scheda bianca).

Motivazione Non consentire espressioni di voto nulli ai sensi delle norme vigenti.

Dipendenze *FS3.4, FS3.5*

Procedimento di configurazione

FU3

Descrizione Il CAE autorizzato deve poter svolgere la configurazione delle sessioni di voto.

Motivazione Consentire la creazione di nuove sessioni di voto, e tutte le sue componenti.

Dipendenze *FS4*

FU3.1

Descrizione Il CAE autorizzato deve poter impostare il luogo, la descrizione, data di inizio e fine della sessione, e le consultazioni che la compongono.

Motivazione Consentire la creazione di nuove sessioni di voto.

Dipendenze *FS4.1*

FU3.2

Descrizione Il CAE autorizzato deve poter inserire nuove consultazioni.

Motivazione Consentire la creazione di una delle componenti di nuove sessioni di voto.

Dipendenze *FS4.1*

FU3.2.1

Descrizione Il CAE autorizzato deve poter impostare la descrizione, il limite di età e il livello di quorum di una data configurazione.

Motivazione Obiettivo della configurazione.

FU3.2.2

Descrizione Il CAE autorizzato deve poter impostare la modalità di voto di una data configurazione.

Motivazione Obiettivo della configurazione.

Dipendenze *FS4.1.1*

FU3.2.3

Descrizione Il CAE autorizzato deve poter impostare la modalità di calcolo del vincitore di una data configurazione.

Motivazione Obiettivo della configurazione.

Dipendenze *FS4.1.2*

FU3.2.4

Descrizione Il CAE autorizzato deve poter inserire le informazioni aggiuntive riguardo la consultazione (lista dei candidati e/o dei partiti oppure quesito per il referendum)

Motivazione Obiettivo della configurazione.

Dipendenze *FS4.1.3*

Procedimento di scrutinio**FU4**

Descrizione Il CAE autorizzato deve essere in grado di attivare il procedimento di scrutinio per una data consultazione.

Motivazione Obiettivo del sistema.

Dipendenze *FS5*

*Altro***FU5**

Descrizione Il gestore del sistema deve poter accedere al log delle operazioni svolte dal sistema.

Motivazione Assicurare il corretto funzionamento del sistema.

Dipendenze *S6*

2.1.2 Requisiti del sistema*Autenticazione***FS1**

Descrizione Il sistema deve svolgere l'autenticazione tramite SPID o CIE.

Motivazione Metodi per l'acquisizione dell'identità digitale già in essere e che rispettano un livello di sicurezza in linea con le norme vigenti.

FS1.1

Descrizione Per il CAE il sistema deve prevedere un ulteriore livello di autenticazione con credenziali fornite dall'amministrazione elettorale in seguito al corso di formazione.

Motivazione Garantire un ulteriore livello di protezione per l'accesso al procedimento di configurazione e/o scrutinio, e garantire che il CAE abbia portato a termine il corso di formazione.

*Procedimento di voto***FS2**

Descrizione Il CAE deve poter inserire il contenuto delle schede tradizionali nel sistema per la consultazione corretta.

Motivazione Assicurare che la preferenza sia per la consultazione corretta.

FS3

Descrizione Il sistema deve consentire all'elettore accesso solo a schede per le consultazioni per cui gode l'elettorato attivo.

Motivazione Assicurare che l'elettore indichi la propria preferenza per le consultazioni corrette.

FS3.1

Descrizione Il sistema ricava le consultazioni attive per l'elettore dalla residenza dello stesso ricavabile dall'identità digitale ottenuta con l'autenticazione tramite SPID o CIE.

Motivazione Assicurare che l'elettore indichi la propria preferenza per le consultazioni corrette.

FS3.2

Descrizione La notifica di corretta registrazione del voto deve essere inviata tramite indirizzo email associata al profilo SPID o CIE con il quale l'elettore ha eseguito l'autenticazione.

Motivazione Confermare all'elettore che il voto è stato registrato correttamente nel sistema di voto centrale.

FS3.3

Descrizione I procedimenti di voto possono essere effettuati solamente entro il periodo di voto.

Motivazione Delimitare il tempo utile in cui è possibile effettuare procedimenti di voto validi.

FS3.4

Descrizione Per ciascuna scheda l'elettore può esprimere un solo voto.

Motivazione Non consentire espressioni di voto nulli ai sensi delle norme vigenti.

FS3.5

Descrizione Per voto valido si intende un voto conforme alla modalità di voto impostata per la consultazione.

Motivazione Non consentire espressioni di voto nulli ai sensi delle norme vigenti ed errate per la consultazione.

Procedimento di configurazione**FS4**

Descrizione Il sistema deve permettere di impostare la configurazione di una sessione di voto.

Motivazione Obiettivo del sistema.

FS4.1

Descrizione Il sistema deve permettere di impostare la configurazione di una consultazione.

Motivazione Obiettivo del sistema.

FS4.1.1

Descrizione Il sistema deve fornire tra le modalità di voto soltanto le opzioni valide.

Motivazione Obiettivo del sistema.

FS4.1.1.1

Descrizione Le opzioni valide per la modalità di voto sono ordinale, categorico, categorico con preferenze e referendum. (Per la definizione precisa fare riferimento al glossario, al punto [1.1](#))

Motivazione Assicurare la corretta impostazione della configurazione.

FS4.1.2

Descrizione Il sistema deve fornire tra le modalità di calcolo del vincitore soltanto le opzioni valide.

Motivazione Assicurare la corretta impostazione della configurazione.

FS4.1.2.1

Descrizione Le opzioni valide per la modalità di calcolo del vincitore sono maggioranza e maggioranza assoluta (Per la definizione precisa fare riferimento al glossario, al punto [1.1](#))

FS4.1.2.2

Descrizione Il sistema deve fornire tra le modalità di voto maggioranza e maggioranza assoluta se e solo se le modalità di voto sono voto ordinale, categorico o categorico con preferenze.

FS4.1.3

Descrizione Il sistema deve permettere l'inserimento delle informazioni aggiuntive riguardo la consultazione.

Motivazione Assicurare la corretta impostazione della configurazione.

FS4.1.3.1

Descrizione Il sistema deve permettere l'inserimento delle liste dei candidati o dei partiti in caso di modalità di voto ordinale o categorico.

Dipendenze *FS4.1.3.3*

FS4.1.3.2

Descrizione Il sistema deve permettere l'inserimento delle liste dei partiti in caso di modalità di voto categorico con preferenze.

Dipendenze *FS4.1.3.3*

FS4.1.3.3

Descrizione Il sistema deve permettere la creazione di nuovi candidati e partiti.

FS4.1.3.4

Descrizione Il sistema deve permettere l'inserimento del quesito in caso di modalità di voto referendum.

FS4.1.4

Descrizione Il sistema deve permettere al CAE autorizzato di confermare le proprie scelte se e solo se ha indicato la descrizione, la modalità di voto, la modalità di calcolo del vincitore, il limite di età, il livello di quorum e la lista dei candidati o dei partiti.

Motivazione Assicurare la corretta impostazione della consultazione.

Procedimento di scrutinio**FS5**

Descrizione Il sistema deve permettere al CAE autorizzato di attivare il procedimento di scrutinio.

Motivazione Obiettivo del sistema.

FS5.1

Descrizione Il procedimento di scrutinio deve essere attivabile solo dopo la terminazione del periodo di voto per la consultazione.

Motivazione Assicurare la corretta registrazione di tutti i voti espressi, e che la consultazione duri effettivamente quanto previsto dal periodo di voto.

FS5.2

Descrizione Il calcolo del vincitore deve essere effettuato secondo le modalità di calcolo del vincitore.

Motivazione Assicurare che il procedimento di scrutinio venga eseguito correttamente.

FS5.3

Descrizione Prima della chiusura del procedimento di scrutinio il sistema deve comunicare i risultati agli OPRU.

Motivazione Assicurarsi che i risultati delle consultazioni arrivino ai destinatari finali.

Dipendenze *S4, SE2*

Altro

FS6

Descrizione Il sistema deve memorizzare la gerarchia dei permessi dei CAE.

Motivazione Assicurare che le operazioni vengano eseguite da CAE autorizzati.

Dipendenze *U2*

FS6.1

Descrizione I permessi del CAE riguardano le consultazioni per i quali ha diritto di scrutinio, e se ha diritto di configurazione delle sessioni di voto e delle sue componenti.

Motivazione Assicurare che le operazioni vengano eseguite da CAE autorizzati.

FS7

Descrizione Il gestore di sistema deve poter inserire requisiti aggiuntivi per una consultazione da gestire in fase di configurazione.

Motivazione Personalizzare le impostazioni per la configurazione.

2.2 Requisiti non funzionali

2.2.1 Requisiti di qualità del software

Q1

Descrizione Il sistema deve essere conforme ai requisiti di usabilità e accessibilità previsti dalla legge.

Motivazione Supporto alle necessità di elettori diversamente abili e con esigenze speciali.

Q1.1

Descrizione Il sistema deve mostrare all'elettore schede elettorali simili a quelle cartacee utilizzate in modalità di voto tradizionali.

Motivazione Rendere agevole il passaggio dalla modalità di voto tradizionale a quella digitale.

Dipendenze *U1.2*

Q2

Descrizione Nel periodo di voto il sistema deve essere funzionante a meno di 2 minuti di downtime.

Motivazione Assicurare il servizio durante il periodo di voto.

Q2.1

Descrizione Nel caso di interruzione accidentale il sistema deve informare immediatamente gli utenti che stanno utilizzando il sistema.

Motivazione Garantire trasparenza nei confronti dell'utente.

Q2.1.1

Descrizione Il sistema deve riprendere il funzionamento dal punto in cui era stata interrotta la sessione.

Motivazione Garantire la facilità d'uso del sistema.

Q2.1.2

Descrizione Il sistema deve prevedere che l'utente si autentichi nuovamente.

Motivazione Garantire l'affidabilità dell'identità dell'utente connesso dopo il periodo di downtime.

2.2.2 Requisiti di sicurezza

Procedimento di voto

S1

Descrizione Il sistema deve rilevare in ogni fase del procedimento di voto eventuali alterazioni dei voti.

Motivazione Preservare la corrispondenza tra voto registrato e la volontà dell'elettore.

S2

Descrizione Nella notifica di conferma di registrazione del voto non deve comparire la preferenza espressa dall'elettore.

Motivazione Preservare la segretezza del voto.

Procedimento di scrutinio

S3

Descrizione Il sistema deve rilevare in ogni fase del procedimento di scrutinio eventuali tentativi di alterazione dei voti.

Motivazione Preservare la corrispondenza tra voti registrati e le volontà degli elettori.

S4

Descrizione La comunicazione dei risultati agli OPRU deve essere eseguita tramite mail PEC.

Motivazione Garantire l'integrità e l'affidabilità delle informazioni inviate.

Altro

S5

Descrizione Il sistema deve tener traccia su un log di tutte le operazioni rilevanti svolte.

Motivazione Garanzia della tracciabilità delle operazioni del sistema.

S5.1

Descrizione Per operazioni rilevanti da documentare nel log si intendono l'accesso e l'uscita di utenti (elettori e CAE), l'effettuazione di procedimenti di voto e di scrutinio, e la creazione di sessioni di voto e relative componenti durante procedimenti di configurazione.

Motivazione Garanzia della tracciabilità delle operazioni del sistema.

Requisiti etici

SE1

Descrizione Il sistema deve garantire segretezza del voto, cioè non deve permettere la riconducibilità della preferenza espressa all'elettore.

Motivazione Obiettivo del sistema.

SE1.1

Descrizione Il dispositivo personale utilizzato per usufruire del sistema non deve mantenere traccia dei dati trattati nella fase di voto.

Motivazione Preservare la segretezza del voto e la riservatezza dei dati utente.

SE1.2

Descrizione Il sistema non deve memorizzare eventuali scelte modificate o cancellate.

Motivazione Preservare la segretezza del voto, ed assicurare la corretta registrazione della preferenza espressa.

SE1.3

Descrizione Prima del procedimento di scrutinio deve essere eseguito un mescolamento dei voti registrati.

Motivazione Non lasciare traccia del momento e dell'ordine di espressione di voto.

SE2

Descrizione I risultati comunicati agli OPRU non devono essere visibili al CAE.

Motivazione Il CAE non può avere un vantaggio nella visualizzazione dei risultati.

Requisiti normativi**SN1**

Descrizione I dati personali dell'utente devono essere trattati in linea con disposizioni in materia di protezione dei dati personali.

Motivazione Mantenere la privacy dei dati gestiti dal sistema.

SN2.1

Descrizione Il sistema deve essere sviluppato rispettando le linee guida AgID per sviluppo software sicuro.

Motivazione Garantire un livello riconosciuto di sicurezza del sistema.

2.2.3 Requisiti degli utenti propedeutici all'utilizzo del sistema**U1**

Descrizione Il CAE deve partecipare ad un corso formativo.

Motivazione Garantire un corretto utilizzo del sistema.

U1.1

Descrizione Il CAE deve essere in grado di inserire il contenuto delle schede tradizionali nel sistema.

Motivazione Garantire il corretto inserimento del contenuto delle schede tradizionali.

U1.2

Descrizione Il CAE deve essere in grado di spiegare all'elettore il procedimento di voto.

Motivazione Garantire la facilità d'uso per l'elettore.

U1.3

Descrizione Il CAE deve essere in grado di svolgere il procedimento di scrutinio senza errori.

Motivazione Garantire la correttezza del procedimento di scrutinio.

U1.4

Descrizione In seguito alla formazione il CAE deve ricevere credenziali da utilizzare nella fase di autenticazione nel ruolo di CAE.

Motivazione Garantire che il CAE abbia portato a termine il corso di formazione.

2.2.4 Requisiti per la fase di testing**T1**

Descrizione Il sistema deve passare da una fase di simulazione.

Motivazione Alpha testing

T1.1

Descrizione La fase di simulazione coinvolge un campione di minimo 1000 elettori.

T1.2

Descrizione La fase di simulazione coinvolge almeno 10 CAE con permessi differenti.

T2

Descrizione Il sistema deve passare da una fase di sperimentazione dopo l'esito positivo della simulazione.

Motivazione Beta testing

T2.1

Descrizione La sperimentazione deve essere svolta in almeno 100 consultazioni in differenti ambiti territoriali, comunali o consolari.

T3

Descrizione Prima di ciascuna tornata elettorale l'amministrazione elettorale, in collaborazione con un organismo indipendente nominato e autorità preposte alla sicurezza cibernetica, devono verificare che il funzionamento del sistema sia in linea con i requisiti.

Motivazione Garantire l'affidabilità del sistema.

Capitolo 3

Progettazione

3.1 Diagramma dei casi d'uso

Il diagramma riportato nella figura 3.1 descrive in modo astratto le funzionalità messe a disposizione dal sistema. Nel punto seguente, per ciascun caso d'uso, vengono presentati i possibili scenari in cui possono trovarsi gli attori coinvolti.

3.1.1 Descrizione degli scenari dei casi d'uso

Per ciascuno scenario di caso d'uso sono specificati i requisiti associati (con riferimento al capitolo 2), lo scopo del caso d'uso, gli attori a cui è associato, pre e post condizioni, la sequenza degli eventi dello scenario, con un livello di elenco per ciascuna variazione rispetto alla sequenza dello scenario genitore, e i trigger scatenanti.

Autenticazione

Requisiti associati *FU1* (2.1.1) con relative dipendenze.

Scopo Verificare la presenza dell'identità digitale dell'utente nel database nazionale.

Attore Utente

Post-condizioni L'utente ha accesso alle diverse funzionalità offerte dal sistema dipendentemente dal suo ruolo.

Sequenza eventi

1. Il sistema richiede all'utente le credenziali SPID o CIE.
2. L'utente inserisce le credenziali SPID o CIE.
3. Se le credenziali sono corrette l'utente risulta autenticato.
 - (a) Se l'autenticazione non ha avuto successo l'utente può ritentare fino a due volte (tre tentativi totali), ripartendo dal punto 1.
4. Se l'utente è sia elettore che CAE, sceglie se entrare nel sistema sotto il ruolo di elettore o CAE.
 - (a) Se l'utente è entrato come CAE, il sistema richiede le credenziali ottenute dopo il corso di formazione (vedi *Autenticazione CAE* in 3.1.1).

Trigger L'ingresso dell'utente nel sistema.

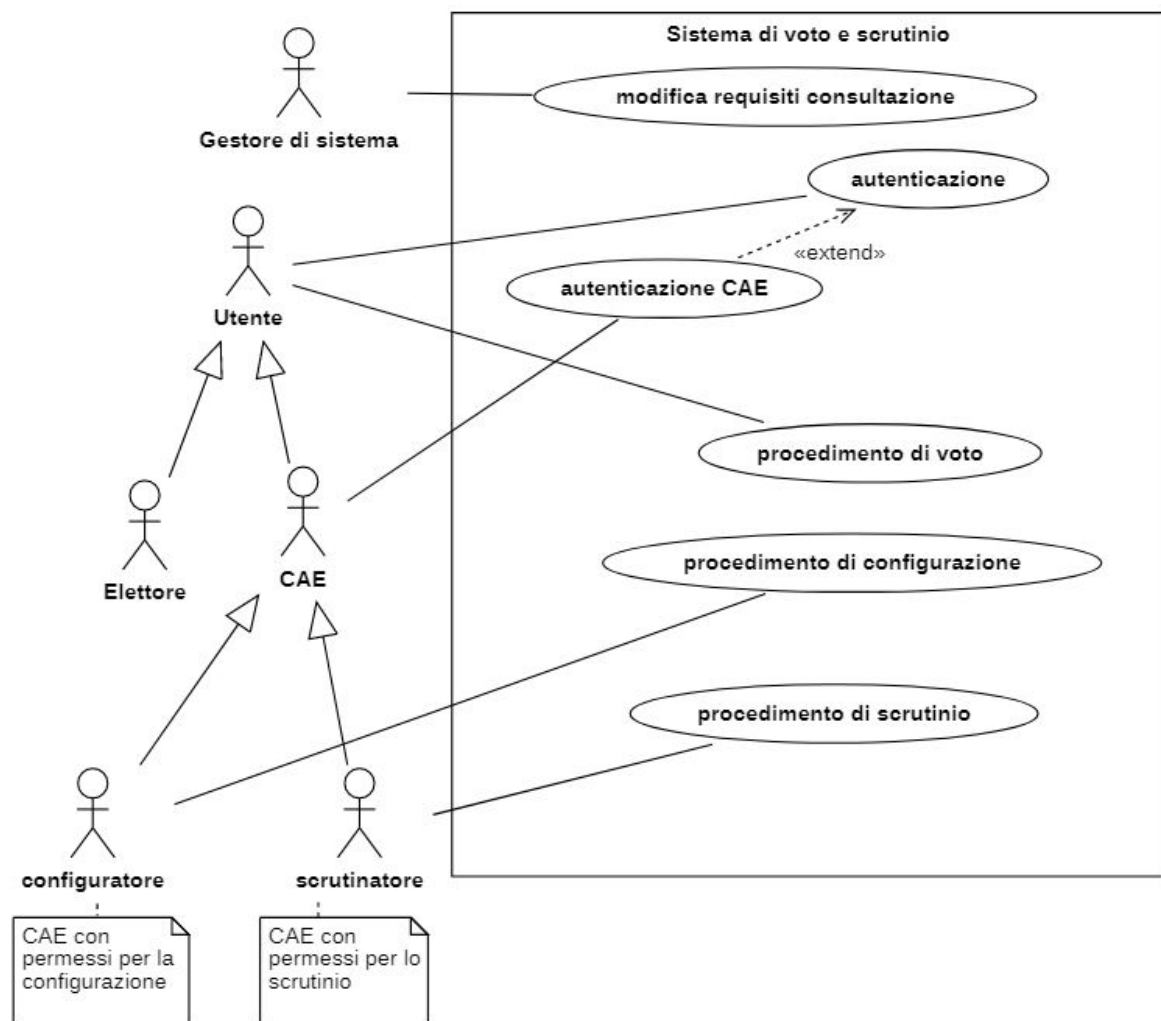


Figura 3.1: Diagramma dei casi d'uso

Autenticazione CAE**Requisiti associati** U1.4 (2.2.3)**Scopo** Verificare che il CAE ha seguito il corso di formazione, e assicurare che il sistema dia la possibilità di svolgere le funzionalità corrette.**Attore** CAE**Post-condizioni** Il CAE ha accesso alle diverse funzionalità offerte dal sistema dipendentemente dai suoi permessi.**Sequenza eventi**

1. Il sistema richiede al CAE le credenziali ottenute al completamento del corso di formazione.
2. Il CAE inserisce le credenziali.
3. Se le credenziali sono corrette il CAE risulta autenticato.
 - (a) Se l'autenticazione non ha avuto successo il CAE può ritentare fino a due volte (tre tentativi totali), ripartendo dal punto 1.

Trigger L'ingresso del CAE nel sistema.**Procedimento di voto****Requisiti associati** FU2 (2.1.1) con relative dipendenze.**Scopo** Permettere la registrazione nel sistema centrale delle preferenze espresse tramite schede elettorali tradizionali e digitali.**Attore** Utente**Pre-condizioni**

- Avvenuta autenticazione
- Il procedimento di voto viene svolto entro il periodo di voto
- Elettore avente almeno l'età limite indicata per la consultazione
- L'elettore non ha svolto altri procedimenti di voto per la consultazione corrente

Post-condizioni La preferenza dell'elettore per la consultazione risulta registrata nel sistema centrale.**Sequenza eventi**

1. Il sistema dichiara il tempo massimo necessario per l'espressione delle preferenze.
2. All'utente vengono mostrate tutte le schede elettorali per la sessione attiva durante il periodo di voto.
3. Per ciascuna scheda elettorale il sistema mostra all'utente le relative informazioni in base alla configurazione della stessa.
4. L'utente esprime la propria preferenza (nel caso del CAE inserisce il contenuto della scheda tradizionale).
5. L'utente conferma la preferenza espressa.
 - (a) Fin quando l'utente non conferma la preferenza inserita lo scenario riprende dal punto 2.
6. Il sistema invia una notifica di conferma di registrazione del voto all'elettore.
 - (a) Non è necessario invio della notifica di conferma di registrazione del voto tradizionale al CAE nel caso di inserimento di schede tradizionali.

Trigger L'utente entra nell'area di voto.

Procedimento di configurazione

Requisiti associati $FU3$ (2.1.1) con relative dipendenze.

Scopo Impostare nuove sessioni e le sue componenti.

Attore Configuratore

Pre-condizioni Avvenuta autenticazione CAE e il CAE ha permessi per la configurazione.

Post-condizioni La sessione risulta configurata, cioè non presenta ambiguità sull'impostazione delle sue componenti.

Sequenza eventi

1. Il sistema dà la possibilità di inserire la descrizione, il luogo, data di inizio e di fine della sessione di voto, e le configurazioni che la compongono.
2. Se necessario il configuratore può decidere di creare una nuova consultazione.
 - (a) Il sistema dà la possibilità di inserire la descrizione, il limite di età e il livello di quorum per la consultazione.
 - (b) Il sistema mostra le possibili modalità di voto.
 - (c) Il configuratore sceglie una delle modalità di voto disponibili.
 - (d) Dipendentemente dalla modalità di voto scelta, il sistema mostra le possibili modalità di calcolo del vincitore
 - (e) Il configuratore sceglie la modalità di calcolo del vincitore.
 - (f) Il configuratore sceglie le ulteriori informazioni da inserire nella scheda elettorale.
 - i. Se la modalità di voto scelta è il referendum il sistema permette di inserire il quesito.
 - ii. Se la modalità di voto scelta è il voto ordinale o categorico, il sistema permette di selezionare i partiti o candidati.
 - iii. Se la modalità di voto è il voto categorico con preferenze, il sistema permette di selezionare i partiti.
 - A. Il configuratore può creare un nuovo candidato.
 - B. Il configuratore può creare un nuovo partito.
 - (g) Fin quando il configuratore non imposta tutte le componenti della configurazione il sistema riprende dal punto 2a.
3. Fin quando il configuratore non imposta tutte le componenti della sessione il sistema riprende dal punto 1.

Trigger Il configuratore entra nell'area di configurazione.

Procedimento di scrutinio

Requisiti associati $FU4$ (2.1.1) con relative dipendenze.

Scopo Eseguire il conteggio dei voti raccolti per una consultazione e la comunicare i risultati agli OPRU.

Attore Scrutinatore, OPRU

Pre-condizioni Avvenuta autenticazione CAE, il CAE ha permessi per lo scrutinio e sono terminati i periodi di voto delle sessioni a cui appartiene la consultazione.

Post-condizioni Il risultato della consultazione e il vincitore sono stati comunicati agli OPRU.

Sequenza eventi

1. Il sistema svolge il conteggio dei voti.
2. Il sistema determina il vincitore dipendentemente dalla modalità di voto della consultazione.
3. Il sistema comunica i risultati della consultazione agli OPRU.

Trigger Lo scrutatore accede all'area di scrutinio.

Modifica requisiti delle configurazioni

Requisiti associati *FS7* (2.1.2)

Scopo Permettere l'aggiornamento o la personalizzazione del sistema con eventuali modifiche nei parametri di configurazione delle sessioni di voto e delle sue componenti.

3.2 Diagramma delle classi

Per una esplosione dei contenuti dei diagrammi illustrati in questa sezione si faccia riferimento ai diagrammi riportati nel capitolo riguardante l'implementazione del sistema, al punto 4.1

3.2.1 Diagramma delle classi di *model*

Il diagramma nella figura 3.2 descrive in modo astratto l'organizzazione del modello del sistema.

3.2.2 Diagramma delle classi di *database*

Il diagramma nella figura 3.3 evidenzia le classi utilizzate per interagire con i dati persistenti secondo il Design Pattern DAO, come esplicitato al punto 4.5.1.

3.2.3 Diagramma delle classi di *controllers-model*

Il diagramma nella figura 3.4 mette in relazione le classi del pacchetto *controllers* e *model*. Questa interazione secondo il Design Pattern MVC viene approfondita al punto 4.5.2. Per chiarezza grafica non sono stati esplicitati i particolari delle associazioni tra le classi dei due pacchetti. Per tutte le associazioni il nome segue la convenzione [nome_classe(iniziale minuscola)], la visibilità è *private*, e la molteplicità lato classi di *model* è 1.

Gli oggetti del modello mantenuti nelle classi che controllano l'interfaccia grafica sono stati utilizzati con due finalità.

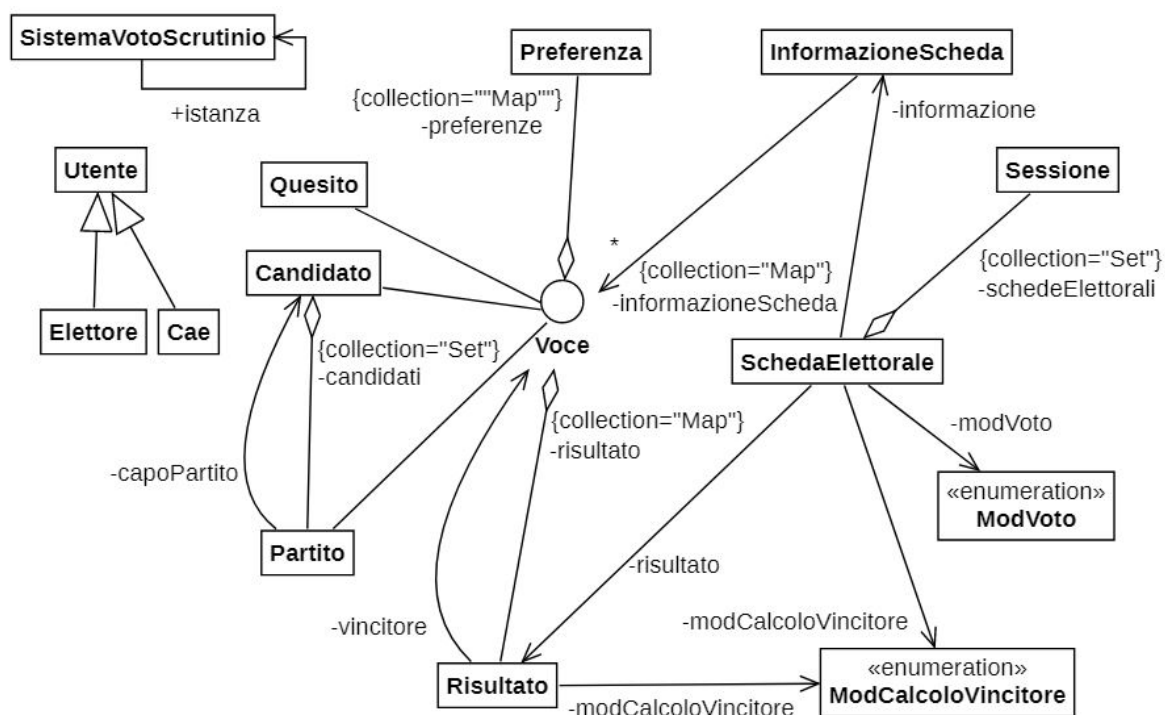
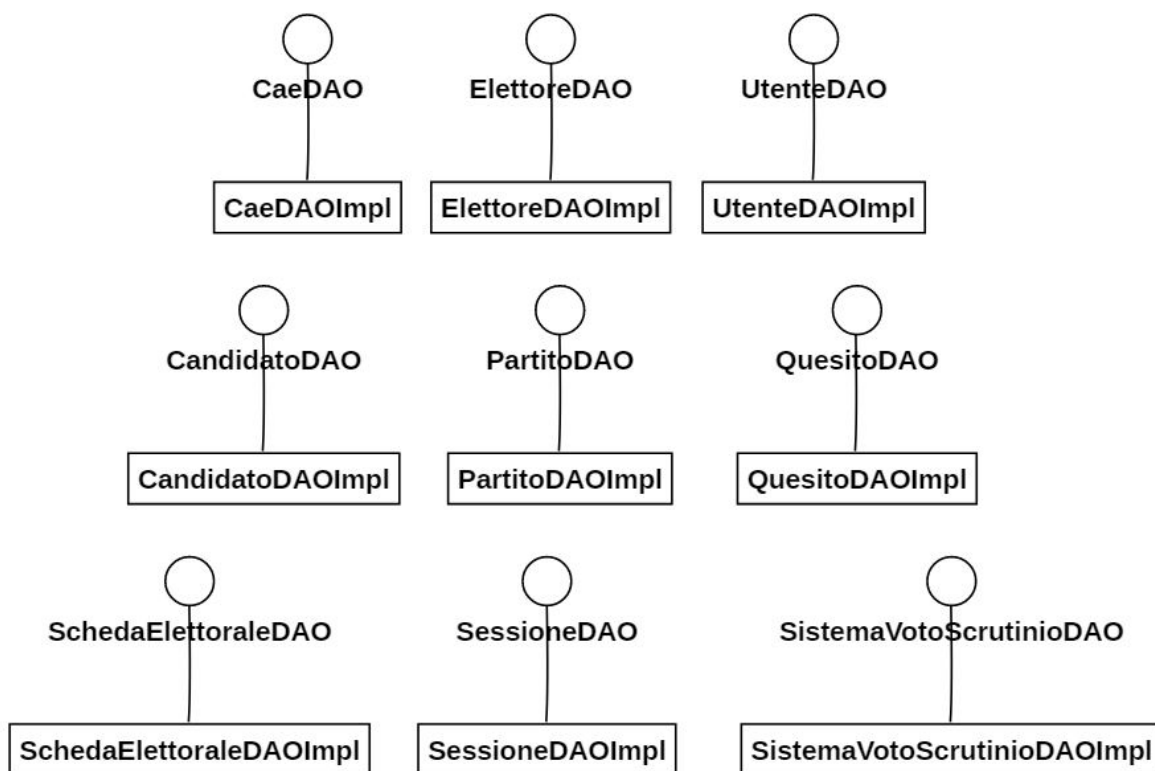
- Dare la possibilità di impostare oggetti a differenti componenti della scena, nel caso in cui la pagina sia costruita in modo incrementale dal controller.

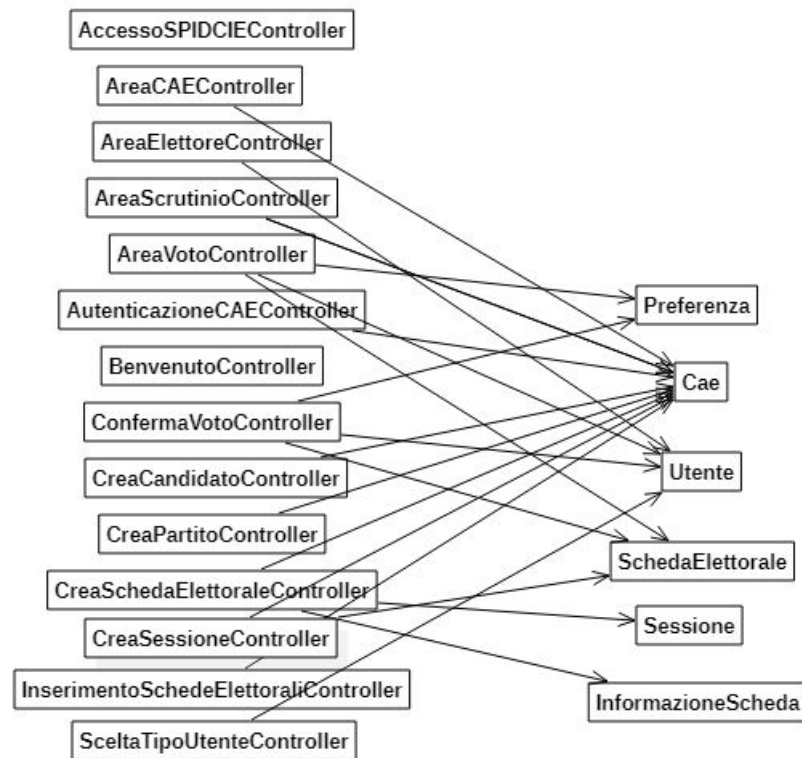
Questo approccio è utile nella situazione in cui differenti casistiche necessitano una diversa interazione con l'utente del sistema. Ad esempio, nell'area di voto la modalità di voto determina come l'elettore esprime la propria preferenza. Similmente, la modalità di voto scelta per una scheda elettorale dal configuratore determina quali scelte presentare per completare l'informazione della scheda in questione.

In tal caso ciascuna parte della scena presentata dinamicamente può accedere all'oggetto del modello nella classe controller.

- Salvare informazioni ottenute da scene precedenti per permettere all'utente di sistema di tornare a pagine visitate in precedenza, senza perdita dell'informazione necessaria per il loro corretto funzionamento.

Questo approccio viene adottato per poter accedere alle informazioni degli utenti (per es. permessi del CAE, luogo di residenza o età dell'elettore), utili a garantire un corretto funzionamento dei casi d'uso associati agli utenti.

Figura 3.2: Diagramma delle classi di *model*Figura 3.3: Diagramma delle classi di *database*

Figura 3.4: Diagramma delle classi di *controllers-model*

3.3 Diagrammi delle attività

Trattandosi di diagrammi redatti in fase di progettazione, le responsabilità raffigurate con le swimlane si riferiscono agli attori presenti nel diagramma dei casi d'uso (3.1). Si è scelto, infatti, di concentrare l'attenzione sulle attività da svolgere, non scendendo in dettaglio sugli oggetti responsabili per l'elaborazione dei dati persistenti e la gestione dell'interfaccia grafica. I messaggi di tipo accept e send indicano interazioni con il database o con l'interfaccia grafica.

Autenticazione

Nel diagramma in figura 3.5 vengono descritti in modo formale gli eventi dello scenario di caso d'uso di autenticazione e autenticazione CAE (3.1.1, 3.1.1).

Procedimento di configurazione

Il diagramma in figura 3.6 ricalca la descrizione dello scenario d'uso procedimento di configurazione (3.1.1). Sono stati aggiunti degli ingrandimenti su alcune delle attività, mostrati nelle figure 3.7 e 3.8. Le attività *crea partito* e *crea candidato* ricalcano la struttura base per la creazione di sessioni e consultazioni, cioè una fase in cui si mostrano le possibili scelte, seguita dalla ricezione dell'impostazione, e la creazione dell'oggetto.

3.4 Diagrammi di sequenza

In questo capitolo dedicato alla progettazione del sistema è stato riportato il diagramma delle classi del progetto, che non è altamente dettagliato. Per questo motivo i diagrammi di sequenza sono riportati nel capitolo sull'implementazione nella sezione 4.3, per permettere un affiancamento al diagramma delle classi di programma.

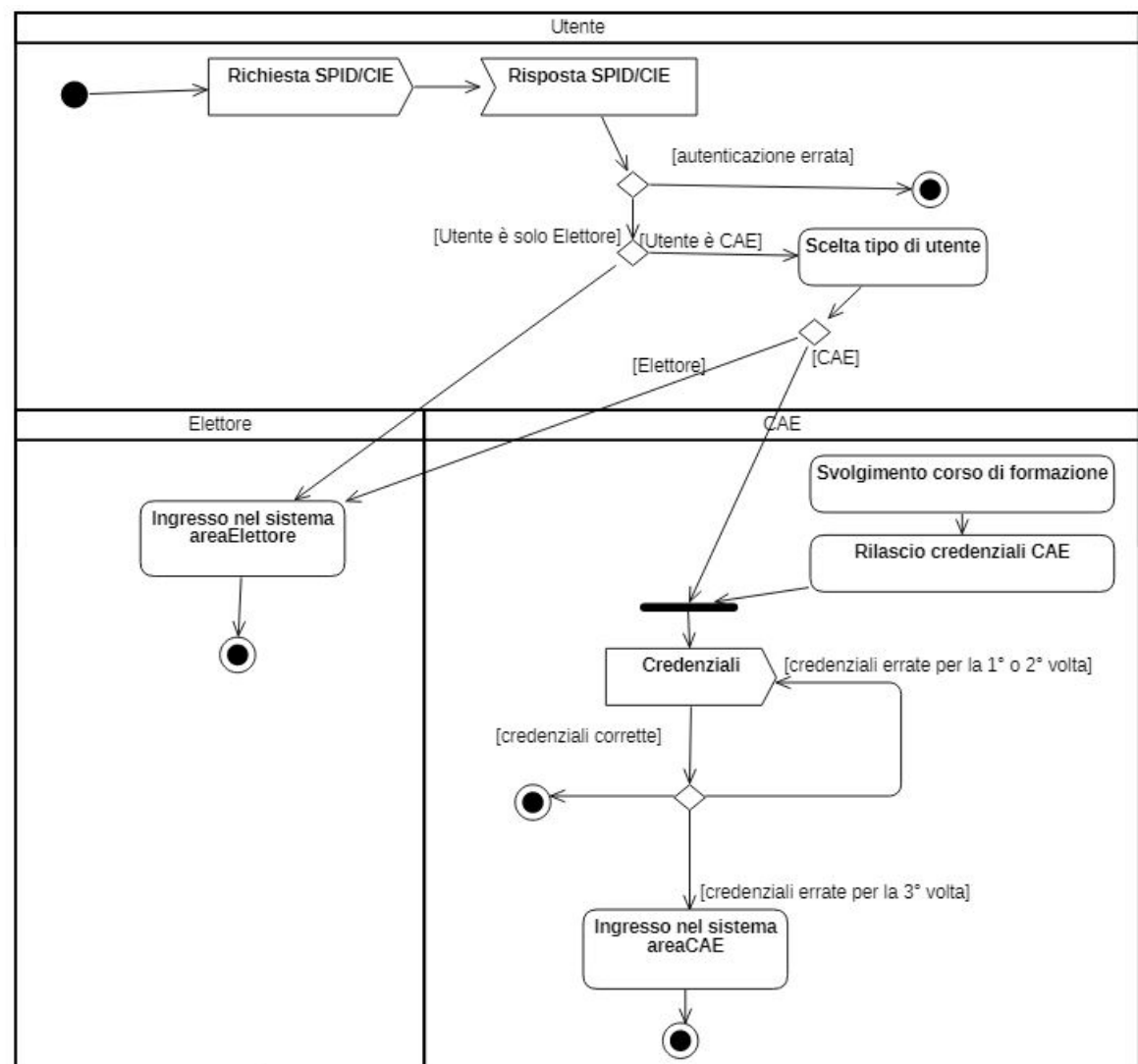


Figura 3.5: Diagramma delle attività dell'autenticazione (inclusa l'autenticazione CAE)

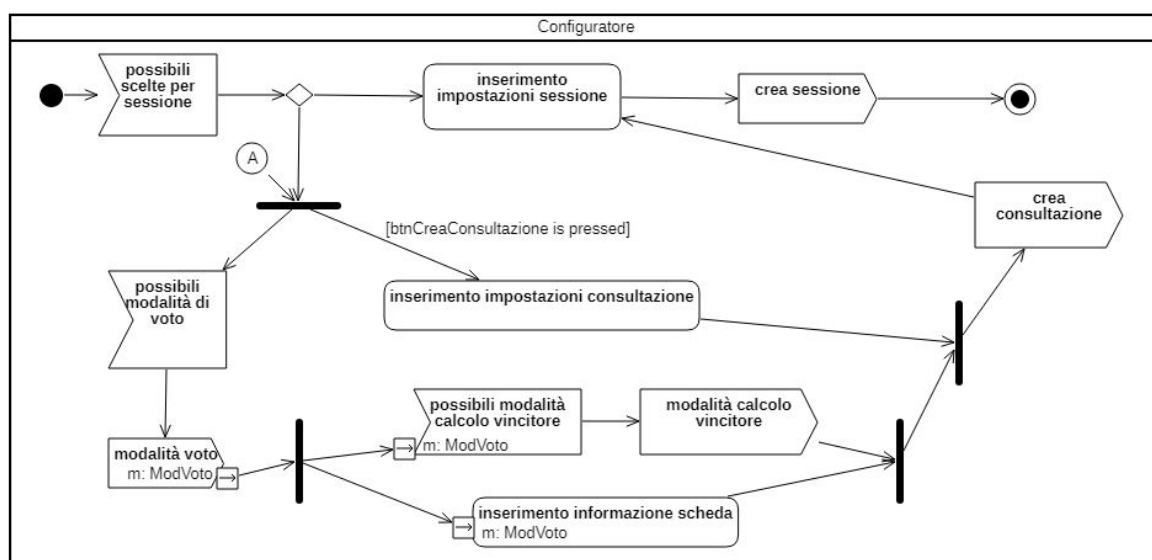


Figura 3.6: Diagramma delle attività del procedimento di configurazione

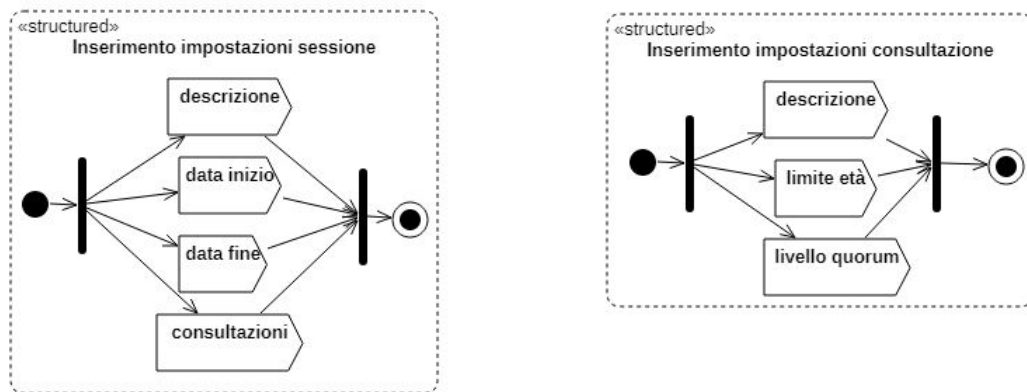


Figura 3.7: Diagramma delle attività del procedimento di voto - particolari *inserimento impostazioni consultazione* e *inserimento impostazioni sessione*

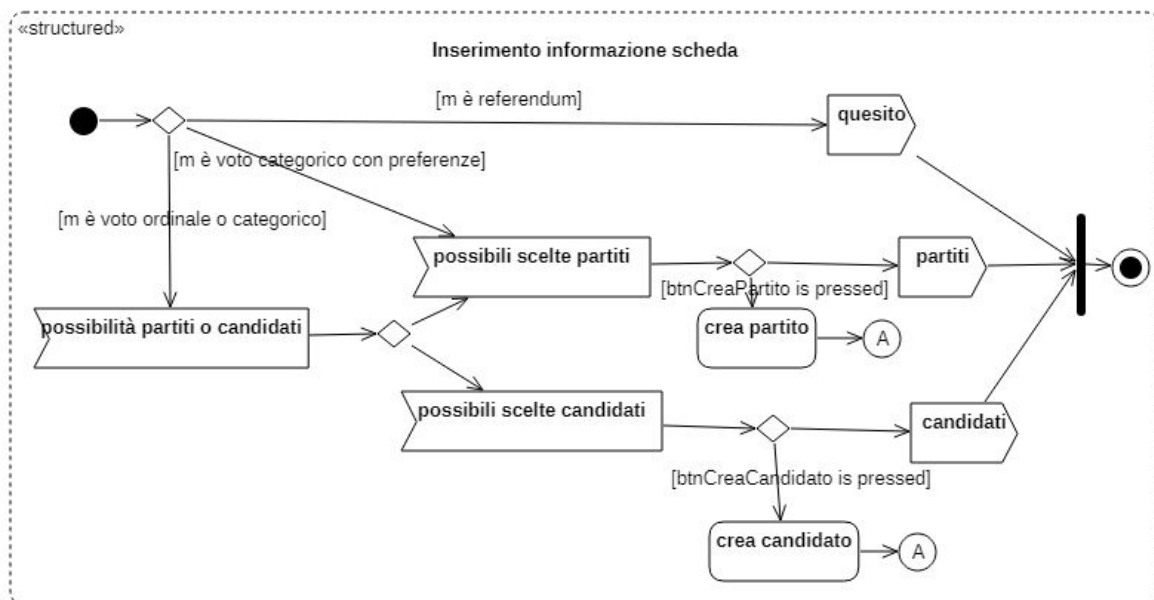


Figura 3.8: Diagramma delle attività del procedimento di voto - particolare inserimento informazione scheda

3.5 Macchine di stato

Come per i diagrammi di sequenza, il livello di dettaglio del diagramma delle classi di programma è necessario per esprimere gli eventi descritti nelle macchine di stato; perciò sono presentati nella sezione [4.4](#).

3.6 Diagramma delle componenti

Com'è possibile notare le tre macro componenti del sistema sono implementate dai tre pacchetti in cui è suddiviso il codice. Inoltre, questa struttura è funzionale per alcuni Design Pattern utilizzati, quali DAO ([4.5.1](#)) e MVC ([4.5.2](#)).

3.7 Diagramma di deployment

L'idea per l'utilizzo del sistema è di prevedere dispositivi dedicati nei luoghi previsti per le votazioni, che hanno già installato il codice necessario per far funzionare il sistema. Nel caso di utenti che vogliono usufruire delle funzionalità del sistema altrove, il codice di SistemaVotoScrutinio va scaricato e installato sul proprio dispositivo. Per questo motivo si prevede sul server centrale un repository, al quale si può accedere remotamente per importare l'eseguibile del sistema. Il sistema funziona come un unicum poiché ha un database centralizzato, mantenuto su un server. Ovviamente è possibile mantenere questi due servizi (database e codice eseguibile) sul medesimo server.

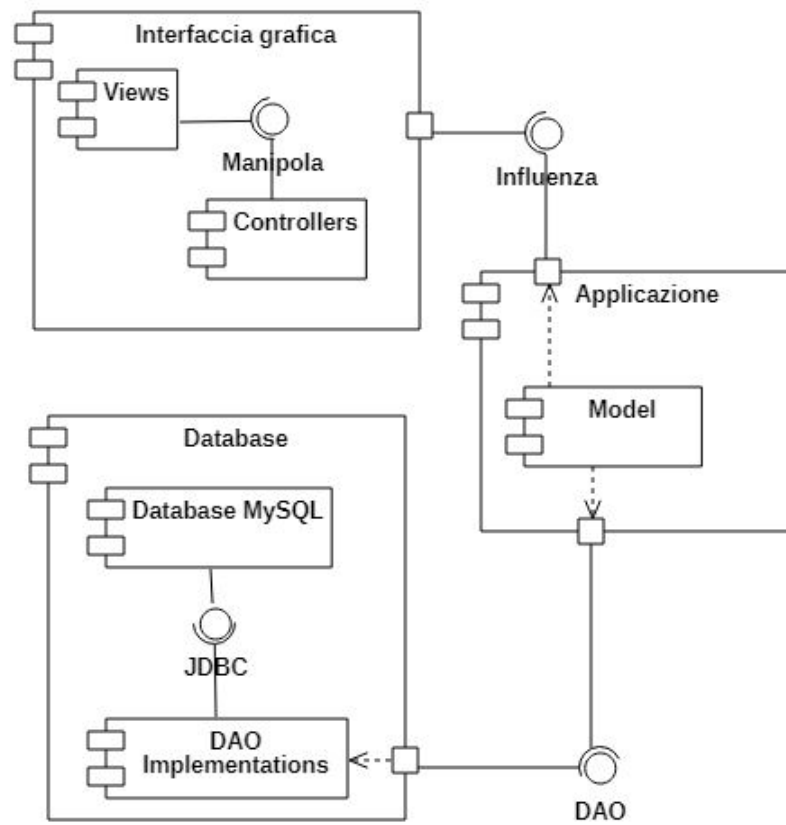


Figura 3.9: Diagramma delle componenti

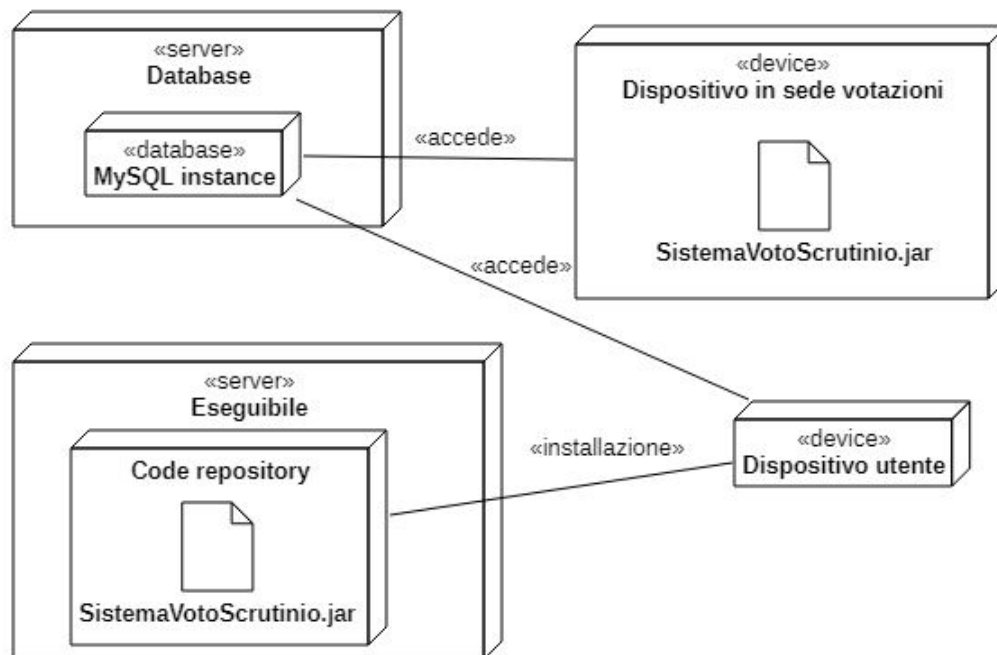


Figura 3.10: Diagramma di deployment

Capitolo 4

Implementazione

4.1 Diagramma delle classi

Seguono i diagrammi delle classi di programma per i pacchetti per i quali si è ritenuta necessaria l'esplosione dei rispettivi diagrammi delle classi di progetto, riportati nella sezione 3.2. Si faccia riferimento a questi ultimi per le relazioni e associazioni tra classi, dal momento che sono stati omessi per non sovraffollare le figure.

4.1.1 Diagramma delle classi di *model*

Le figure 4.1 - 4.14 mostrano gli attributi e le signature dei metodi previsti per ciascuna classe del pacchetto *model*.

Le classi sono presentate secondo ordine alfabetico.

Cae
-scrutinatore: Boolean -configuratore: Boolean
«constructor»+Cae(email: String, scrutatore: boolean, configuratore: boolean) «constructor»+Cae(email: String) «constructor»+Cae(u: Utente) +esiste(): boolean +isConfiguratore(): boolean +isScrutinatore(): boolean -passwordCorretta(password: String): boolean +accesso(password: String): boolean +uscita(): void +scrutinio(s: SchedaElettorale): boolean +comunicaRisultato(s: SchedaElettorale): String +creaSchedaElettorale(s: SchedaElettorale): void +isCae(): boolean +getSchedeElettorali(): SchedaElettorale[*] -configuratorePerScheda(s: SchedaElettorale): boolean +getSchedeElettoraliConElettori(): SchedaElettorale[*] +toString(): String +equals(obj: Object): boolean +hashCode(): int

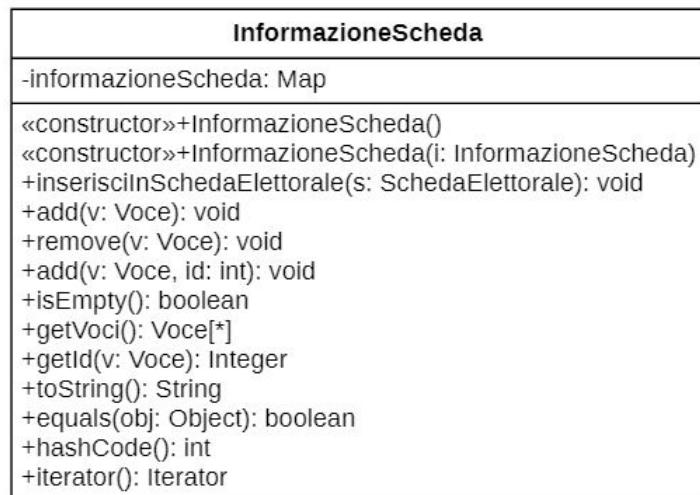
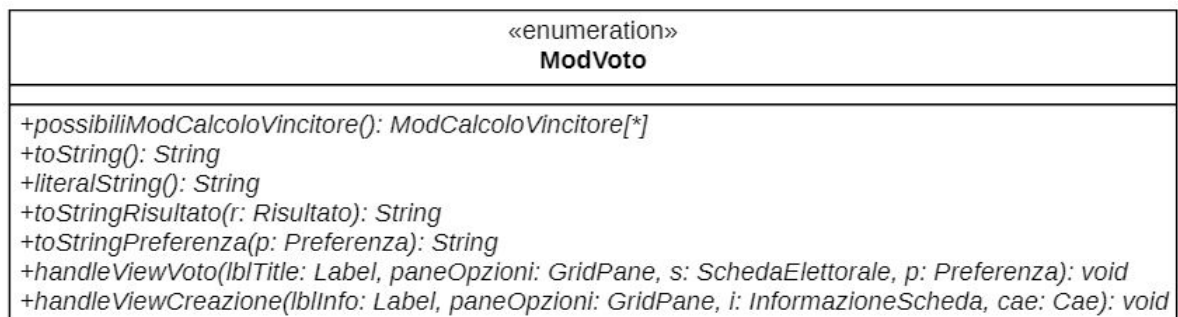
Figura 4.1: Classe *Cae*

Candidato
-codiceFiscale: String -nome: String -cognome: String
«constructor»+Candidato(codiceFiscale: String, nome: String, cognome: String) «constructor»+Candidato(codiceFiscale: String) +getCodiceFiscale(): String +getNome(): String +getCognome(): String +crea(): void +elimina(): void +esiste(): boolean +inserisciInSchedaElettorale(s: SchedaElettorale): int +toString(): String +equals(obj: Object): boolean +hashCode(): int

Figura 4.2: Classe *Candidato*

Elettore
-codiceFiscale: String -luogoResidenza: String
«constructor»+Elettore(email: String) «constructor»+Elettore(email: String, codiceFiscale: String, luogoesidenza: String) «constructor»+Elettore(u: Utente) «constructor»+Elettore(u: Utente, codiceFiscale: String, luogoResidenza: String) +crea(): void +registrato(): boolean +registrato(email: String): boolean +esprimiPreferenza(s: SchedaElettorale, p: Preferenza): void +comunicaPreferenzaEspressa(): void +preferenzaEspressa(s: SchedaElettorale): void +getSessioniInCorso(): Sessione[*] +getSchedeElettorali(): SchedaElettorale[*] +getSchedeElettoraliSenzaPreferenza(): SchedaElettorale[*] -autorizzato(etaMin: int): boolean -dataDiNascitaDaCodiceFiscale(cf: String): Date +getCodiceFiscale(): String +getLuogoResidenza(): String +isCae(): boolean +toString(): String +equals(o: Object): boolean +hashCode(): int

Figura 4.3: Classe *Elettore*

Figura 4.4: Classe *InformazioneScheda*Figura 4.5: Classe *ModCalcoloVincitore*Figura 4.6: Classe *ModVoto*

Partito
-nome: String
«constructor»+Partito(nome: String) «constructor»+Partito(nome: String, capoPartito: Candidato, candidati: Candidato[*]) +getNome(): String +getCapoPartito(): Candidato +getCandidati(): Candidato[*] +crea(): void +esiste(): boolean +elimina(): void +inserisciInSchedaElettorale(s: SchedaElettorale): int +iterator(): Iterator +toString(): String +equals(obj: Object): boolean +hashCode(): int

Figura 4.7: Classe *Partito*

Preferenza
-preferenze: Map
«constructor»+Preferenza() «constructor»+Preferenza(p: Preferenza) +add(v: Voce, n: Integer): void +remove(v: Voce): void +bianca(): boolean ~getPreferenze(): Entry<Voce[*] +toString(): String +equals(obj: Object): boolean +hashCode(): int +iterator(): Iterator

Figura 4.8: Classe *Preferenza*

Quesito
~id: int -quesito: String
«constructor»+Quesito(quesito: String) «constructor»+Quesito(id: int, quesito: String) «constructor»+Quesito(id: int) +getId(): int +crea(): void +elimina(): void +esiste(): boolean +inserisciInSchedaElettorale(s: SchedaElettorale): int +getQuesito(): String +toString(): String +equals(obj: Object): boolean +hashCode(): int

Figura 4.9: Classe *Quesito*

Risultato
-risultato: Map -numeroVotanti: int
«constructor»+Risultato(modCalcoloVincitore: ModCalcoloVincitore, numeroVotanti: int, voci: Voce[*]) «constructor»+Risultato(risultato: Risultato) +getNumeroVotanti(): int +getVincitore(): Voce +calcolaVincitore(): void ~getResultato(): Map +add(v: Voce, ris: int): void +toString(): String +equals(obj: Object): boolean +hashCode(): int +iterator(): Iterator

Figura 4.10: Classe *Risultato*

SchedaElettorale
-id: int -descrizione: String -limiteEta: int -quorum: int
«constructor»+SchedaElettorale(descrizione: String, i: InformazioneScheda, limiteEta: int, mV: ModVoto, mCV: ModCalcoloVincitore, quorum: int) «constructor»+SchedaElettorale(id: int, descrizione: String, i: InformazioneScheda, limiteEta: int, mV: ModVoto, mCV: ModCalcoloVincitore, quorum: int) «constructor»+SchedaElettorale(id: int) +getId(): int +getDescription(): String +getLimiteEta(): int +getModVoto(): ModVoto +getInformazione(): InformazioneScheda +getModCalcoloVincitore(): ModCalcoloVincitore +getResultato(): Risultato +getQuorum(): int +esiste(): boolean +crea(): void +elimina(): void +esprimiPreferenza(p: Preferenza): void +getPreferenze(): Entry<Voce[*] +getSessioni(): Sessione[*] +numeroElettoriTotali(): int +numeroElettoriEffettivi(): int +scrutinio(): boolean +toStringRisultato(): String +toString(): String +equals(obj: Object): boolean +hashCode(): int

Figura 4.11: Classe *SchedaElettorale*

Sessione
-id: int -descrizione: String -inizio: Date -fine: Date -luogo: String
«constructor»+Sessione(id: int) «constructor»+Sessione(id: int, descrizione: String, inizio: Date, fine: Date, schedeElettorali: SchedaElettorale[], luogo: String) «constructor»+Sessione(descrizione: String, inizio: Date, fine: Date, schedeElettorali: SchedaElettorale[], luogo: String) +getDescrizione(): String +getDataInizio(): Date +getDataFine(): Date +getId(): int +getLuogo(): String +getSchedeElettorali(): SchedaElettorale[] +inserisciElettore(e: Elettore): void +crea(): void +elimina(): void +iterator(): Iterator +toString(): String +equals(obj: Object): boolean +hashCode(): int

Figura 4.12: Classe *Sessione*

SistemaVotoScrutinio
-logger: BufferedWriter
«constructor»-SistemaVotoScrutinio() <u>+getIstanza(): SistemaVotoScrutinio</u> +log(s: String): void +getNumeroElettori(): int +getElettori(): Elettore[] +getSessioni(): Sessione[] +getSchedeElettorali(): SchedaElettorale[] +getPartiti(): Partito[] +getCandidati(): Candidato[] +getCandidatiSenzaPartito(): Candidato[] +getQuesiti(): Quesito[] +toString(): String +equals(obj: Object): boolean +hashCode(): int

Figura 4.13: Classe *SistemaVotoScrutinio*

Utente
#email: String
«constructor»+Utente(email: String) +getEmail(): String +crea(password: String): void +registrato(): boolean +elimina(): void -passwordCorretta(password: String): boolean +accesso(password: String): boolean +uscita(): void +isCae(): boolean +toString(): String +equals(o: Object): boolean +hashCode(): int

Figura 4.14: Classe *Utente*

4.1.2 Diagramma delle classi di *database*

Questi diagrammi mostrano i metodi previsti per le classi DAO utili a gestire la persistenza dati. I metodi di queste classi vengono invocati opportunamente dai metodi delle classi in *model*.

Tutte le classi, proprio perché addette all'interazione con il database, mettono a disposizione almeno i metodi *crea*, *elimina* ed *esiste*.

CaeDAOImpl
<pre> «constructor»+CaeDAOImpl() -getConnection(): Connection +getCae(email: String): Cae +crea(c: Cae, password: String): void +getCae(): Cae[*] +aggiornaPassword(c: Cae, password: String): void +esiste(c: Cae): boolean +elimina(c: Cae): void -hash(s: String): String +passwordCorretta(c: Cae, password: String): boolean +isScrutinatore(c: Cae): boolean +isConfiguratore(c: Cae): boolean +getSchedeElettorali(c: Cae): SchedaElettorale[*] +inserisciSchedaElettorale(c: Cae, s: SchedaElettorale): void </pre>

Figura 4.15: Classe che implementa *CaeDAO*

CandidatoDAOImpl
<pre> «constructor»+CandidatoDAOImpl() -getConnection(): Connection +elimina(c: Candidato): void +crea(c: Candidato): void +inserisciInSchedaElettorale(s: SchedaElettorale, c: Candidato): int +getCandidato(codiceFiscale: String): Candidato +esiste(c: Candidato): boolean </pre>

Figura 4.16: Classe che implementa *CandidatoDAO*

ElettoreDAOImpl
<pre> «constructor»+ElettoreDAOImpl() -getConnection(): Connection +elimina(e: Elettore): void +crea(e: Elettore): void +getElettori(): Elettore[*] +esiste(e: Elettore): boolean +codiceFiscale(e: Elettore): String +getSessioni(e: Elettore): Sessione[*] +segnaVoceComeNonEsprimibile(e: Elettore, s: SchedaElettorale): void +preferenzaEspressa(e: Elettore, s: SchedaElettorale): boolean +getElettore(email: String): Elettore </pre>

Figura 4.17: Classe che implementa *ElettoreDAO*

PartitoDAOImpl
«constructor»+PartitoDAOImpl() -getConnection(): Connection +elimina(p: Partito): void +getPartiti(): Partito[*] +crea(p: Partito): void +inserisciInSchedaElettorale(s: SchedaElettorale, p: Partito): int +getPartito(nome: String): Partito +esiste(p: Partito): boolean -getCandidati(nome: String): Candidato[*] +getCapoPartito(p: Partito): Candidato

Figura 4.18: Classe che implementa *PartitoDAO*

QuesitoDAOImpl
«constructor»+QuesitoDAOImpl() -getConnection(): Connection +elimina(q: Quesito): void +crea(q: Quesito): int +inserisciInSchedaElettorale(s: SchedaElettorale, q: Quesito): int +getQuesito(id: int): Quesito +esiste(q: Quesito): boolean

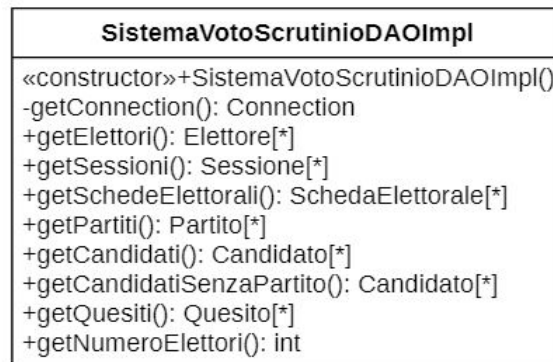
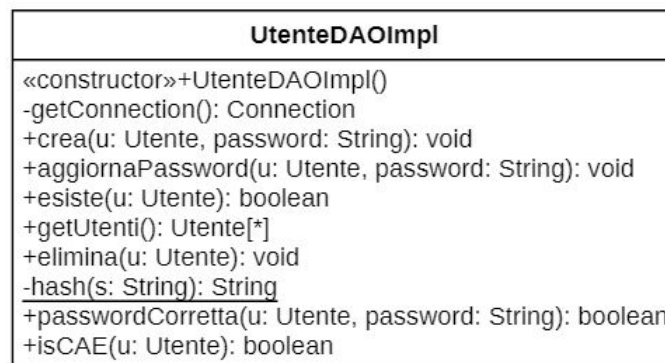
Figura 4.19: Classe che implementa *QuesitoDAO*

SchedaElettoraleDAOImpl
«constructor»+SchedaElettoraleDAOImpl() -getConnection(): Connection +elimina(s: SchedaElettorale): void +crea(s: SchedaElettorale): int +esprimiPreferenza(idVoceInSchedaElettorale: int, preferenza: int): void +getSchedaElettorale(id: int): SchedaElettorale -getInformazione(idScheda: int): InformazioneScheda +getSessioni(s: SchedaElettorale): Sessione[*] +getNumeroElettoriTotali(s: SchedaElettorale): int +getNumeroElettoriEffettivi(s: SchedaElettorale): int +getPreferenze(s: SchedaElettorale): Entry<Voce[*] +esiste(s: SchedaElettorale): boolean

Figura 4.20: Classe che implementa *SchedaElettoraleDAO*

SessioneDAOImpl
«constructor»+SessioneDAOImpl() -getConnection(): Connection +elimina(s: Sessione): void +crea(c: Sessione): int +inserisciElettore(s: Sessione, e: Elettore): void +inserisciSchedaElettorale(c: Sessione, s: SchedaElettorale): void -getSchedeElettorali(id: int): SchedaElettorale[*] +getSessione(id: int): Sessione +esiste(s: Sessione): boolean

Figura 4.21: Classe che implementa *SessioneDAO*

Figura 4.22: Classe che implementa *SistemaVotoScrutinioDAO*Figura 4.23: Classe che implementa *UtenteDAO*

4.2 Vincoli

4.2.1 Vincoli OCL

Le espressioni dei vincoli OCL da applicare al diagramma delle classi del pacchetto *model* vengono riportate di seguito, per una maggiore chiarezza.

I vincoli che coinvolgono l'interazione con il database, dal momento in cui non sono esprimibili con la sintassi OCL, sono stati espressi adottando in parte linguaggio informale. In fase di implementazione questi vincoli sono stati espressi formalmente in linguaggio JML, come descritto al punto 4.2.2.

Non sono stati esplicitati i vincoli per i metodi ereditati, in particolare *toString*, *equals*, *hashCode* e *iterator*.

Per la sintassi utilizzata per la gestione del tipo di dato Map si fa riferimento all'[estensione ideata da Willink](#).

Cae

```

{context Cae::Cae(email: String, scrutatore: boolean, configuratore: boolean)
  pre: email <> null}
{context Cae::Cae(email: String)
  pre: email <> null}
{context Cae::Cae(email: String)
  pre: oggetto con chiave primaria email esiste in tabella cae in database}
{context Cae::Cae(u: Utente)
  pre: oggetto con chiave primaria u.email esiste in tabella cae in database}
{context Cae::isConfiguratore(): boolean
  post: result = self.configuratore}
{context Cae::isScrutatore(): boolean
  post: result = self.scrutatore}
{context Cae::passwordCorretta(password: String): boolean

```

```

    pre: password <> null}
{context Cae::accesso(password: String): boolean
    pre: password <> null}
{context Cae::scrutinio(s: SchedaElettorale): boolean
    pre: s <> null post: self.scrutinatore implies result = false}
{context Cae::creaSchedaElettorale(s: SchedaElettorale)
    pre: s <> null post: result = self.configuratore}
{context Cae::isCae(): boolean
    post: result = true}
{context Cae::getSchedeElettorali(): Set(SchedaElettorale)
    post: result <> null}
{context Cae::getSchedeElettoraliConElettori(): Set(SchedaElettorale)
    post: result <> null}

```

Candidato

```

{context Candidato
    inv: self.codiceFiscale <> null and self.nome <> null and self.cognome <> null}
{context Candidato::Candidato(codiceFiscale: String, nome: String, cognome: String)
    pre: codiceFiscale <> null and nome <> null and cognome <> null}
{context Candidato::Candidato(codiceFiscale: String)
    pre: codiceFiscale <> null}
{context Candidato::Candidato(codiceFiscale: String)
    pre: oggetto con chiave primaria codiceFiscale esiste in tabella candidato in database}
{context Candidato
    inv: self.codiceFiscale.matches('[A-Z]{6}[0-9]{2}[A-Z][0-9]{2}[A-Z][0-9]{3}[A-Z]')}
{context Candidato::getCodiceFiscale(): String
    post: result = self.codiceFiscale}
{context Candidato::getNome(): String
    post: result = self.nome}
{context Candidato::getCognome(): String
    post: result = self.cognome}

```

Elettore

```

{context Elettore inv: self.codiceFiscale <> null and luogoResidenza <> null}
{context Elettore
    inv: self.codiceFiscale.matches('[A-Z]{6}[0-9]{2}[A-Z][0-9]{2}[A-Z][0-9]{3}[A-Z]')}
{context Elettore::Elettore(email: String)
    pre: email <> null}
{context Elettore::Elettore(email: String)
    pre: oggetto con chiave primaria email esiste in tabella elettore in database}
{context Elettore::Elettore(email: String, codiceFiscale: String, luogoResidenza: String)
    pre: email <> null and codiceFiscale <> null and luogoResidenza <> null}
{context Elettore::Elettore(u: Utente)
    pre: oggetto con chiave primaria u.email esiste in tabella elettore in database}
{context Elettore::Elettore(u: Utente, codiceFiscale: String, luogoResidenza: String)
    pre: codiceFiscale <> null and luogoResidenza <> null}
{context Elettore::getCodiceFiscale(): String post: result = self.codiceFiscale}
{context Elettore::getLuogoResidenza(): String post: result = self.luogoResidenza}
{context Elettore::esprimiPreferenza(s: SchedaElettorale, p: Preferenza): boolean
    pre: s <> null and s <> null}
{context Elettore::getSessioniInCorso(): Set(Sessione)
    post: result <> null}
{context Elettore::getSchedeElettorali(): Set(SchedaElettorale)
    post: result <> null
    and result ->forall(s | Sessione.allInstances() -> forall(s | s.fine.after(now)))}
{context Elettore::getSchedeElettoraliSenzaPreferenza(): Set(SchedaElettorale)
    post: result <> null}

```

```

    and result ->forall(s | Sessione.allInstances() -> forall(s | s.fine.after(now)))
    and result->forall(s | s.getPreferenze() -> isEmpty())}
{context Elettore::autorizzato(etaMin: Integer): boolean}
{context Elettore::isCae(): boolean post: result = false}

```

InformazioneScheda

```

{context InformazioneScheda inv: self.informazioneScheda <> null}
{context InformazioneScheda::inserisciInSchedaElettorale(s: SchedaElettorale)
  pre: s <> null
  post: self.informazione.keys() ->
    forall(v | v@pre = -1 implies v = self.informazione.id)}
{context InformazioneScheda::add(v: Voce)
  post: self.informazioneScheda.keys() -> includes(v)}
{context InformazioneScheda::add(v: Voce, id: Integer)
  pre: oggetto con chiave primaria email esiste in informazionesscheda elettore in database
  post: self.informazioneScheda.keys() -> includes(v) and informazioneScheda.at(v) = id}
{context InformazioneScheda::remove(v: Voce)
  post: self.informazioneScheda.keys() -> includes(v)}

```

ModCalcoloVincitore

```

{context ModCalcoloVincitore::literalString(): String
  post: result <> null}
{context ModCalcoloVincitore:: calcoloVincitore(r: Risultato): Voce
  pre:r <> null}

```

ModVoto

```

{context ModVoto:: possibiliModCalcoloVincitore(): Sequence(ModCalcoloVincitore)
  post: result <> null}
{context ModVoto::literalString(): String
  post: result <> null}
{context ModVoto::toStringRisultato(r: Risultato): String
  pre: r <> null post: result <> null}
{context ModVoto::toStringPreferenza(p: Preferenza): String
  pre: p <> null post: result <> null}
{context ModVoto::handleViewVoto(s: SchedaElettorale, p: Preferenza)
  pre: s <> null and p <> null}
{context ModVoto::handleViewCreazione(s: SchedaElettorale, p: Preferenza)
  pre: s <> null and p <> null}

```

Partito

```

{context Partito inv: self-nome <> null and self.candidati <> null
  and self.capoPartito <> null}
{context Partito inv: candidati -> includes(capoPartito)}
{context Partito
  inv: Partito.allInstances ->
    one(p | Candidato.allInstances -> forall(c | p.candidati -> includes(c)))}
{context Partito::Partito(nome: String)
  pre: nome <> null}
{context Partito::Partito(nome: String)
  pre: oggetto con chiave primaria nome esiste in database}
{context Partito::Partito(nome: String, capoPartito: Candidato, candidati: Sequence(Candidato))
  pre: nome <> null and candidati <> null}
{context Partito::getNome(): String
  post result = self.nome}
{context Partito::getCapoPartito(): Candidato
  post result = self.capoPartito}

```

```
{context Partito::getCandidati(): Set(Candidato)
  post result <> null}
{context Partito::inserisciInSchedaElettorale(s: SchedaElettorale): boolean
  pre: s <> null}
```

Preferenza

```
{context Preferenza inv: self.preferenze <> null}
{context Preferenza::add(v: Voce, id: Integer)
  post: self.preferenze.keys() -> includes(v) and preferenze.at(k) = id}
{context Preferenza::remove(v: Voce)
  post: self.preferenze.keys() -> includes(v)}
{context Preferenza::bianca(): boolean
  post: self.preferenze -> isEmpty()}
{context Preferenza::getPreferenze(): Set(Tuple(Voce, Integer))
  post: result <> null}
```

Quesito

```
{context Quesito inv: self.quesito <> null}
{context Quesito::Quesito(String quesito)
  post: self.id = -1}
{context Quesito::getId(): Integer
  post: result = self.id}
{context Quesito::getQuesito(): Integer
  post: result = self.quesito}
{context Quesito::crea()
  post: self.id@pre=-1 implies self.id > 0}
{context Quesito::inserisciInSchedaElettorale(s: SchedaElettorale): Integer
  pre: s <> null}
```

Risultato

```
{context Risultato inv: self.risultato <> null and self.modCalcoloVincitore <> null}
{context Risultato::Risultato(modCalcoloVincitore: ModCalcoloVincitore,
  numeroVotanti: Integer, voci: Sequence(Voce))
  pre: voci <> null}
{context Risultato::getNumeroVotanti(): Integer
  post: result = self.numeroVotanti}
{context Risultato::getVincitore(): Integer
  post: self.vincitore <> null and result = self.vincitore}
{context Risultato::getRisultato(): Map(Voce, Integer)
  post: result = self.risultato}
{context Risultato::calcolaVincitore
  post: self.vincitore <> null}
{context Risultato::add(v: Voce, n: Integer)
  pre self.risultato.keys() -> includes(v)
  post: self.preferenze.keys() -> includes(v)
  and self.preferenze.at(v) = self.preferenze.at(v)@pre + n}
```

SchedaElettorale

```
{context SchedaElettorale inv: self.descrizione <> null
  and self.limiteEta>=18 and self.limiteEta<=30
  and self.informazione <> null and self.risultato <> null
  and self.modVoto <> null and self.modCalcoloVincitore <> null
  and self.quorum >= 0 and self.quorum <= 100}
{context SchedaElettorale::SchedaElettorale(id: Integer)
  pre: oggetto con chiave primaria id esiste in tabella schedaelettorale in database}
{context SchedaElettorale::SchedaElettorale(descrizione: String,
```

```

    informazione: InformazioneScheda, limiteEta: Integer, modVoto: ModVoto,
    modCalcoloVincitore: ModCalcoloVincitore, quorum: Integer)
    post: self.id = -1}
{context SchedaElettorale::getId(): Integer
    post: result = self.id}
{context SchedaElettorale::getDescrizione(): String
    post: result = self.descrizione}
{context SchedaElettorale::getModvoto(): ModVoto
    post: result = self.modVoto}
{context SchedaElettorale::getModCalcoloVincitore(): ModCalcoloVincitore
    post: result = self.modCalcoloVincitore}
{context SchedaElettorale::getRisultato(): Risultato
    post: result = self.risultato}
{context SchedaElettorale::getQuorum(): Integer
    post: result > 0}
{context SchedaElettorale::esprimiPreferenza(p: Preferenza)
    pre: p <> null}
{context SchedaElettorale::getPreferenze(): Sequence(Tuple(Voce, Integer))
    post: result <> null}
{context SchedaElettorale::getSessioni(): Set(Sessione)
    post: result <> null}
{context SchedaElettorale::numeroElettoriTotali(): Integer
    post: result > 0}
{context SchedaElettorale::numeroElettoriEffettivi(): Integer
    post: result > 0}
{context SchedaElettorale::scrutinio(): boolean
    post Sessione.allInstances ->
        forall(s | s.schedeElettorali -> contains(s) implies s.fine.after(now))
        implies result = false}
{context SchedaElettorale::toStringRisultato(): String
    post: result <> null}

```

Sessione

```

{context Sessione
    inv: self.descrizione <> null and self.inizio <> null and self.fine <> null
    and self.schedeElettorali <> null and self.luogo <> null}
{context Sessione inv: inizio.getTime() < fine.getTime()}
{context Sessione::Sessione(descrizione: String, inizio: Date, fine: Date,
    schedeElettorali: Sequence(SchedaElettorale), luogo: String)
    pre: descrizione <> null and luogo <> null}
{context Sessione::Sessione(id: Integer)
    pre: oggetto con chiave primaria id esiste in tabella sessione in database}
{context Sessione::getDescrizione()
    post: result = self.descrizione}
{context Sessione::getLuogo()
    post: result = self.luogo}
{context Sessione::getDataInizio()
    post: result = self.inizio}
{context Sessione::getDataFine()
    post: result = self.fine}
{context Sessione::getId()
    post: result = self.id}
{context Sessione::getSchedeElettorali
    post: result <> null}
{context Sessione::inserisciSchedaElettorale(s: SchedaElettorale)
    pre: s <> null}
{context Sessione::inserisciElettore(e: Elettore)
    pre: e <> null}

```

```
{context Sessione::crea()
  post: self.id@pre=-1 implies self.id > 0}
```

SistemaVotoScrutinio

```
{context SistemaVotoScrutinio inv: istanza <> null and logger <> null}
{context SistemaVotoScrutinio inv: SistemaVotoScrutinio.allInstances -> size() <= 1}
{context SistemaVotoScrutinio::getIstanza(): SistemaVotoScrutinio
  post: result <> null}
{context SistemaVotoScrutinio::log(s: String)
  pre: s <> null}
{context SistemaVotoScrutinio::getNumeroElettori(): Integer
  post: result > 0}
{context SistemaVotoScrutinio::getElettori(): Set(Elettore)
  post: result <> null}
{context SistemaVotoScrutinio::getSessioni(): Set(Sessione)
  post: result <> null}
{context SistemaVotoScrutinio::getSchedeElettorali(): Set(SchedaElettorale)
  post: result <> null}
{context SistemaVotoScrutinio::getPartiti(): Set(Partito)
  post: result <> null}
{context SistemaVotoScrutinio::getCandidati(): Set(Candidato)
  post: result <> null}
{context SistemaVotoScrutinio::getCandidatiSenzaPartito(): Set(Candidato)
  post: result <> null
  and result -> forall(c | Partito.allInstances() -> forall(p | p.candidati -> excludes(c)))}
{context SistemaVotoScrutinio::getQuesiti(): Set(Quesito)
  post: result <> null}
```

Utente

```
{context Utente inv: self.email <> null}
{context Utente::Utente(email: String)
  pre: email <> null}
{context Utente::getEmail(): String
  post: result = self.email}
{context Utente::passwordCorretta(password: String)
  pre: password <> null}
{context Utente::accesso(password: String)
  pre: password <> null}
```

4.2.2 Vincoli JML

Segue la mappatura in JML di un sottinsieme dei vincoli OCL specificati nella sezione 4.2.1. Gli esempi riportati sono di tipologie differenti, per permettere di applicare le medesime regole di sintassi a tutti i vincoli espressi sul diagramma delle classi di *model*.

Invariante

```
public class Partito{
  [...]
  /*invariant (\exists Candidato c; candidati.contains(c); c.equals(capoPartito)) @*/
  [...]
}

public class SchedaElettorale{
  private /*@spec_public@*/ int id;
  private /*@spec_public non_null@*/ String descrizione;
  private /*@spec_public@*/ int limiteEta;
  private /*@spec_public non_null@*/ InformazioneScheda informazione;
```



```

    private /*@spec_public non_null@*/ ModVoto modVoto;
    private /*@spec_public non_null@*/ ModCalcoloVincitore modCalcoloVincitore;
    private /*@spec_public@*/ int quorum;
    private /*@spec_public@*/ Risultato risultato;
    /*@invariant (limiteEta >= 18 && limiteEta <= 30) && (quorum >= 0)&&(quorum <= 100) @*/
    [...]
}

```

Pre e post condizioni

```

public class Cae{
    [...]
    /*@ensures \result != null @*/
    private boolean passwordCorretta(String password);
    [...]
    /*@ensures also \result == true @*/
    public boolean isCae();
    [...]
}

public class InformazioneScheda{
    [...]
    /*@requires s != null@*/
    /*@ensures (\forallall Voce v; informazione.getKeys().contains(v); informazione.get(v)!=-1) @*/
    public void inserisciInSchedaElettorale(SchedaElettorale s);
    [...]
}

public class Preferenza{
    [...]
    /*@requires v != null@*/
    /*@ensures (\exists Voce voce; preferenze.getKeys().contains(voce); v.equals(voce)
        && preferenze.get(v) == n @*/
    public void add(Voce v, Integer n);
    [...]
}

public class Quesito{
    [...]
    /*@ensures \old(id) == -1 ==> id>0 @*/
    public void crea();
    [...]
}

```

Pre-condizioni che coinvogono dati persistenti

```

public class Candidato{
    /*@ensures codiceFiscale != null && esiste(codiceFiscale)@*/
    public Candidato(String codiceFiscale) {...}
    [...]
    /*@requires email != null@*/
    private /*@pure@*/ boolean esiste(String codiceFiscale){
        CandidatoDAO cDAO = new CandidatoDAOImpl();
        return cDAO.esiste(codiceFiscale);
    }
    [...]
}

```

4.3 Diagrammi di sequenza

Sono stati riportati i diagrammi di sequenza per i casi in cui si ritiene interessante mostrare come sono stati utilizzati i metodi appartenenti alle classi. In particolare questi diagrammi evidenziano quali responsabilità sono stati attribuiti alle classi.

I messaggi *lost* e *found* sono stati utilizzati per modellare l'interazione con l'interfaccia grafica e i dati persistenti. Nello specifico, quelli associati a classi di *controllers* rappresentano dati ottenuti dall'interazione con l'utente, e mostrati all'utente. Fa eccezione il metodo *load*, utilizzato per modellare il caricamento della scena di *view* associata al controller che riceve tale messaggio. I messaggi associati a classi di *database* sono indicazione di interazione con il database.

Procedimento di voto

Il diagramma di sequenza è suddiviso in porzioni, raffigurate rispettivamente in 4.24, 4.25, 4.26 e 4.27. In linea con la sequenza di eventi descritti nel rispettivo caso d'uso, il diagramma delle sequenze mostra una prima fase di ingresso nell'areaElettore, seguita dal reindirizzamento verso la pagina della consultazione scelta. Segue la gestione della conferma della preferenza inserita. Nel caso in cui viene premuto il bottone per la conferma si procede con la registrazione della preferenza espressa. Al termine della registrazione della preferenza, il sistema riporta l'utente nell'ambiente 'base' rispettivamente per il CAE o per l'Elettore.

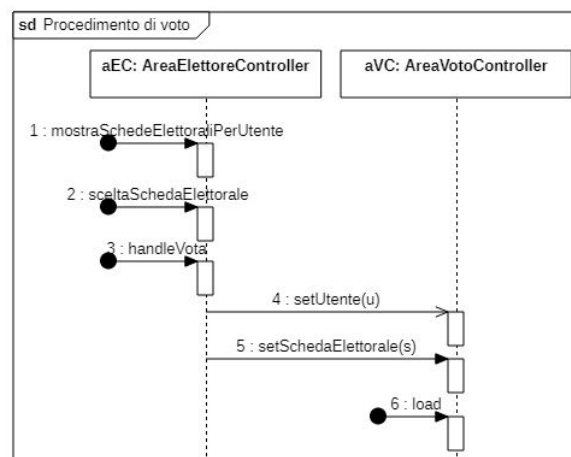


Figura 4.24: Diagramma di sequenza del procedimento di voto - parte 1

Procedimento di scrutinio

Anche in questo caso è stato ricalcato lo scenario di caso d'uso.

Per motivi di leggibilità non è stato approfondito come l'oggetto *m* di tipo *ModCalcoloVincitore* assorbe il metodo *calcolaVincitore(r)*.

4.4 Macchine di stato

SchedaElettorale

La macchina di stato nella figura 4.29 è stata riportata per mostrare che un oggetto di tipo *SchedaElettorale* può non avere associato un risultato valorizzato. Il risultato della consultazione associata alla scheda elettorale ha significato soltanto se è stato effettuato il procedimento di scrutinio (durante il quale viene invocato il metodo *scrutinio*), e se sono terminati tutti i periodi di voto delle sessioni in cui è contenuta la scheda elettorale, in linea con il requisito *FS5.1* (2.1.2).

Quesito & SchedaElettorale & Sessione

Per oggetti di questi tipi l'impostazione dell'attributo *id* è legato al modo in cui vengono inseriti nel database. Difatti, nelle omonime tabelle di riferimento, gli elementi hanno il campo *id* con numerazione

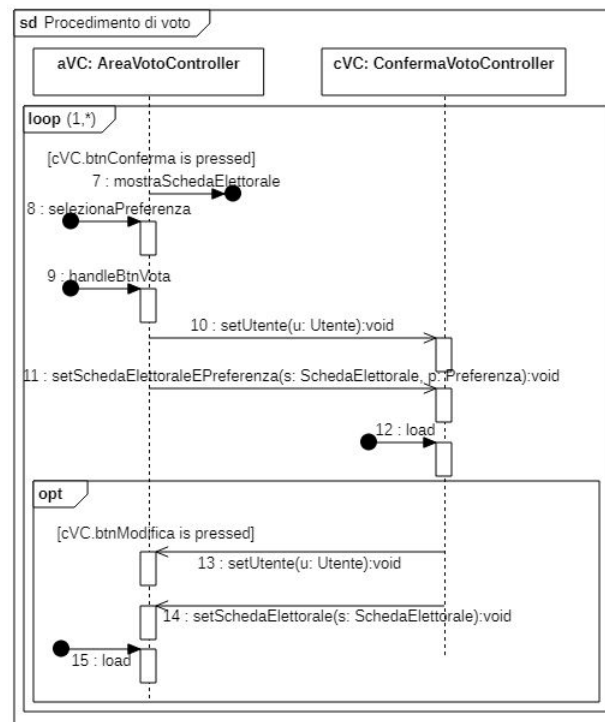


Figura 4.25: Diagramma di sequenza del procedimento di voto - parte 2

incrementale. Dunque nell'istanziamento, che precede l'inserimento nel database (vedi 4.5.1), non è possibile impostare l'id corretto. Perciò questa operazione viene svolta dopo l'inserimento nel database, responsabilità assegnata al metodo *crea*.

Entry di `InformazioneScheda.informazioneScheda`

Si è messo in evidenza il comportamento di una singola entry della mappa *informazioneScheda*, attributo della classe *InformazioneScheda*. Infatti, come spiegato per gli oggetti trattati nel punto precedente, l'istanziamento e l'inserimento nel database avvengono in due fasi distinte, dunque può avvenire che una entry esista all'interno dell'oggetto *InformazioneScheda*, ma che il suo valore non corrisponda ad una chiave primaria valida nel database.

4.5 Descrizione Design Pattern utilizzati

4.5.1 DAO

Il pacchetto *database* contiene le classi che gestiscono l'interazione con il database. Nello specifico il nome delle interfacce che prevedono i metodi per interagire con il database seguono la convenzione di nomenclatura `[nome_classe_model] [DAO]`. Ciascuna interfaccia è implementata dalle classi con nome `[nome_interfaccia_DAO] [Impl]`. I metodi ritenuti necessari per inserire dati e restituire risultati di interrogazioni sul database sono riportati nel diagramma 4.1.2.

Il modello è stato implementato in modo tale che possano esistere oggetti che non hanno corrispettivo nel database. Una volta creati, cioè memorizzati in modo persistente, è possibile passare dalla sola chiave primaria delle tabelle di riferimento per istanziare oggetti che rispecchiano le informazioni mantenute nel database. Ciò viene fatto mettendo a disposizione un costruttore che ha come argomento l'attributo corrispondente alla chiave primaria per l'oggetto, e che inizializza gli altri campi con i valori estratti dal database. In questo modo la "costruzione" del singolo oggetto avviene una volta soltanto con l'obiettivo di renderlo persistente.

4.5.2 MVC

La directory *resources/views* contiene i file fxml gestiti dai relativi controller nel pacchetto *controllers*. Il pacchetto *model*, invece, contiene il modello del sistema. Il Design Pattern in questione è stato

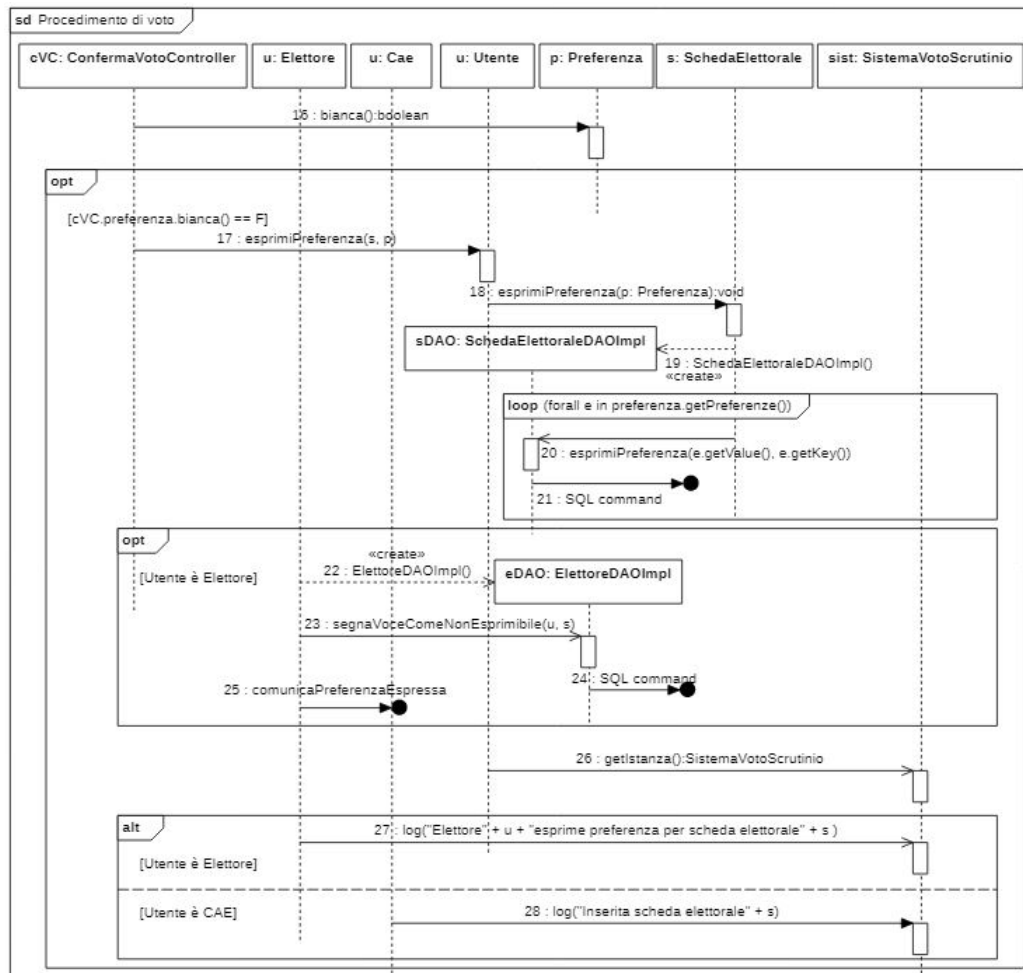


Figura 4.26: Diagramma di sequenza del procedimento di voto - parte 3

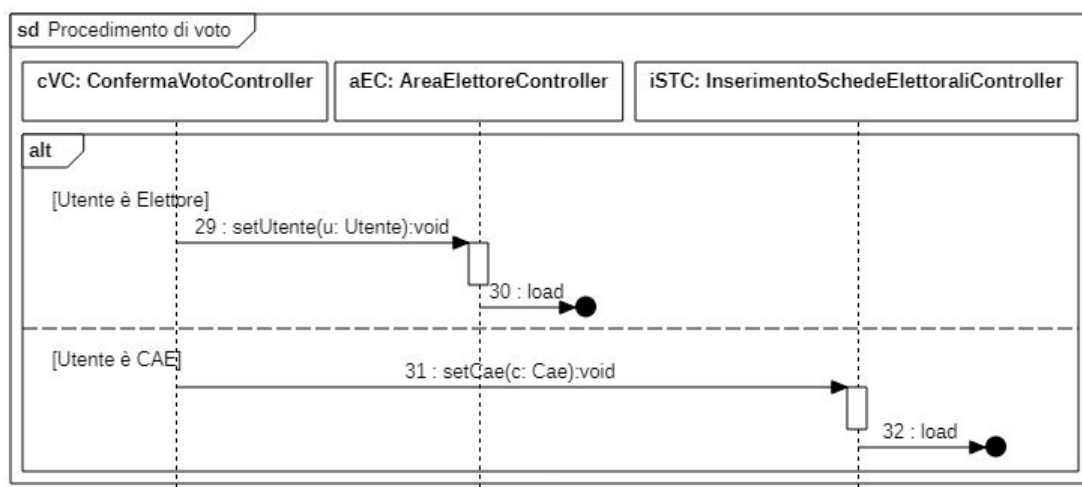


Figura 4.27: Diagramma di sequenza del procedimento di voto - parte 4

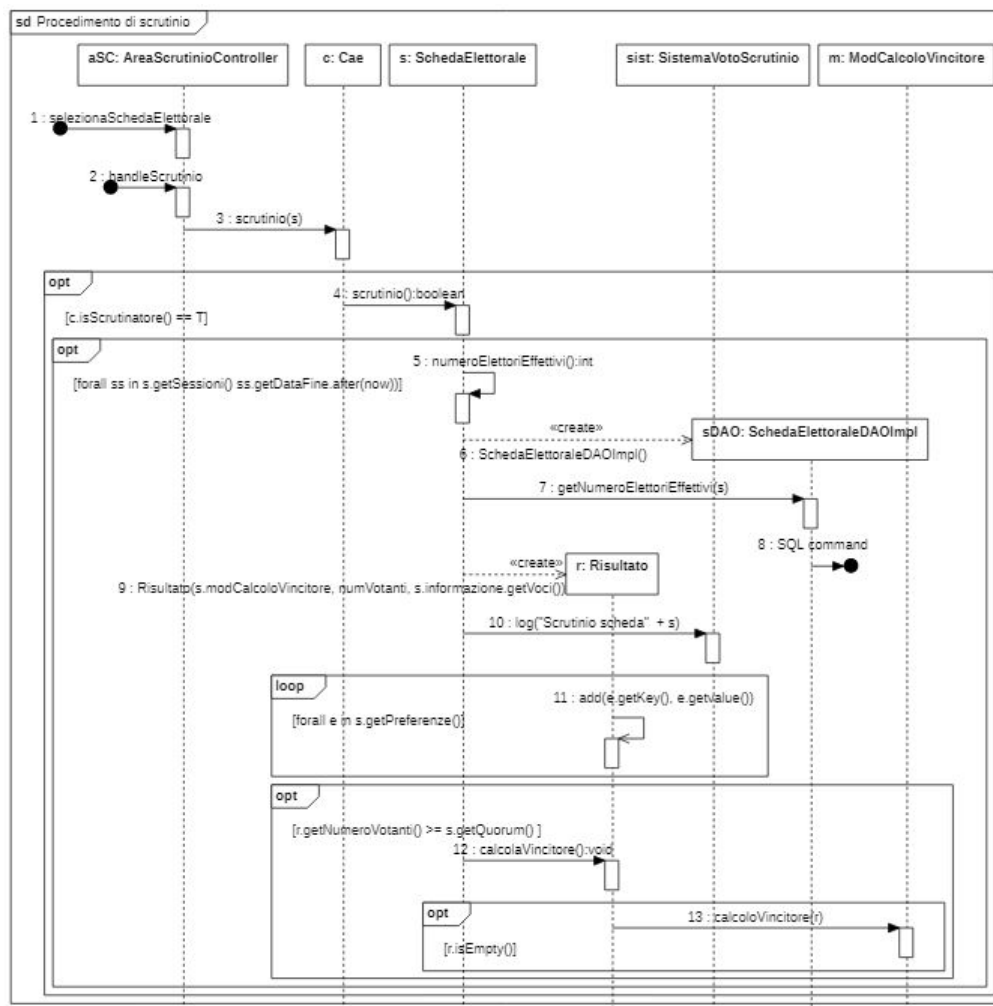


Figura 4.28: Diagramma di sequenza del procedimento di scrutinio

implementato mettendo in relazione i controller con i file fxml. I controller, difatti, contengono i metodi per gestire e modificare la struttura delle scene delineata nei file fxml. I controller, poi, interagiscono con il modello in quanto gestiscono le scene avendo nota la struttura del modello, e hanno come attributi oggetti del modello, come descritto nel diagramma 3.2.3.

Fa eccezione la classe *ModVoto* all'interno della quale sono stati implementati metodi legati all'interfaccia grafica, per permettere di sfruttare il polimorfismo, come discusso in più dettaglio nella sezione 4.6.

4.5.3 Singleton

L'oggetto di tipo *SistemaVotoScrutinio* implementa questo Design Pattern. È stata fatta questa scelta dato che funge il ruolo di information expert, mettendo a disposizione metodi per avere una visione di insieme dei dati persistenti e per fare operazioni di logging. In altri termini, detiene le informazioni generali concernenti il sistema. Visto che il sistema è uno soltanto dal momento che è una proiezione del database sottostante, ha senso prevedere un solo oggetto che funge da informatore e da reporter delle azioni svolte nel file di log.

4.5.4 Observer

Questo Design Pattern è stato utilizzato nell'implementazione dell'interfaccia grafica in tutti quei casi fosse necessario che un elemento della scena mutasse in funzione dell'impostazione di altri elementi. Per fare ciò sono stati implementati "al volo" oggetti di tipo *ChangeListener*, che, come suggerisce il nome, specificano come reagire alla modifica delle proprietà dell'oggetto. L'implementazione ha sfruttato interfacce e metodi messi a disposizione dalle librerie di JavaFX.

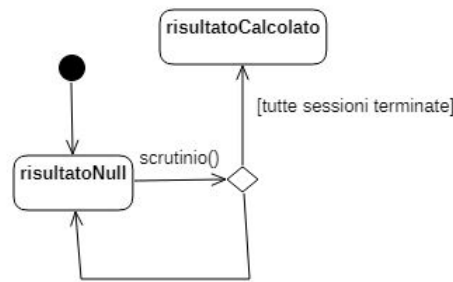
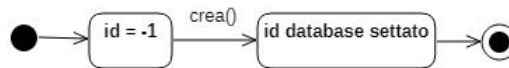
Figura 4.29: Macchina di stato per oggetti di tipo *SchedaElettorale*

Figura 4.30: Macchina di stato per oggetti con impostazione dell'id successiva all'istanziamento

Si elencano gli elementi e le relative scene per cui è stato adottato questo approccio.

areaVoto

- *handleVotoOrdinale.choiceBox* per la gestione delle voci da mostrare
- *handleVotoOrdinale.choiceBox* per la gestione della preferenza scelta
- *handleVotoCategorico.choiceBox* per la gestione della preferenza scelta
- *handleVotoCategoricoConPref.choiceBoxPartiti* per impostare l'insieme dei candidati tra cui è possibile scegliere
- *handleVotoCategoricoConPref.choiceBoxPartiti* per la gestione della preferenza scelta
- *handleVotoCategoricoConPref.choiceBoxCandidati* per la gestione della preferenza scelta
- *handleVotoReferendum.toggleGroup* per la gestione della preferenza scelta

areaCreaPartito

- *initialize.checkBox* per ottenere i candidati scelti

areaCreaSchedaElettorale

- *initialize.choiceModVoto* per gestire l'impostazione degli altri elementi in base alla modalità di voto scelta
- *handleSceltaCandidatiPartiti.toggleGroup* intercettare la scelta per permettere la scelta di partiti oppure di candidati
- *handleInformazioneSchedaCandidati.checkBox* per la gestione dei candidati scelti
- *handleInformazioneSchedaPartiti.checkBox* per la gestione dei partiti scelti
- *handleInformazioneSchedaReferendum.checkBox* per la gestione dell'inserimento del quesito

areaCreaSessione

- *initialize.dataInizio* per impostare la data scelta per l'inizio della sessione come la prima data di fine sessione disponibile
- *initialize.checkBox* per la gestione delle consultazioni scelte

4.6 Discussione scelte high cohesion - low coupling

In questa sezione si motivano scelte non ovvie in relazione all'accoppiamento e alla coesione che comportano.

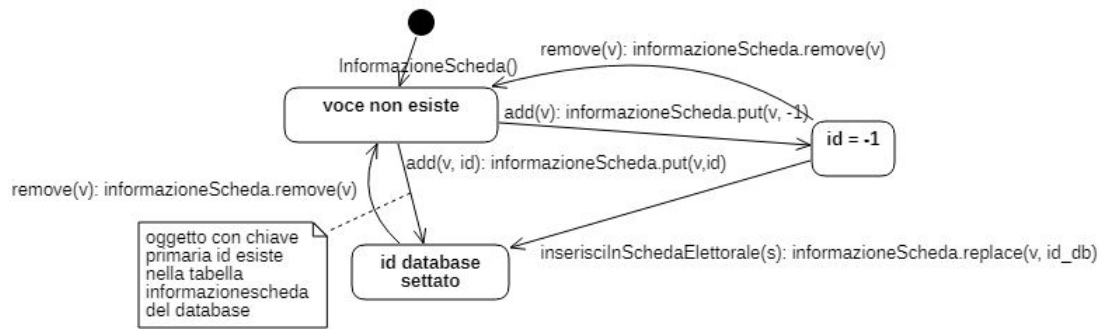


Figura 4.31: Macchina di stato per una singola entry della mappa *informazioneScheda* in oggetti tipo *InformazioneScheda*

Si è scelto di inserire i metodi *handleViewVoto* e *handleViewCreazione* nella classe *ModVoto* per permettere di sfruttare il polimorfismo dell'enumerazione. Questi metodi determinano come rendere rispettivamente l'area di voto e l'area di configurazione delle schede elettorali nell'interfaccia grafica. Questo approccio, considerato che i metodi sono inseriti in una classe del pacchetto *model*, causa una bassa coesione. È da notare che i metodi in *ModVoto* non implementano direttamente la gestione delle view; piuttosto invocano metodi statici nel pacchetto *controllers* che dettagliano il comportamento dell'interfaccia grafica. Si è ritenuto opportuno, in questo caso, mancare di coesione per evitare numerosi switch case nelle classi controllers coinvolte. Questa scelta facilita anche l'eventuale inserimento di differenti modalità di voto: è sufficiente implementare i metodi dell'interfaccia data nell'enumerazione *ModVoto*, e non è necessario ricercare nei controller le situazioni in cui l'interfaccia grafica si comporta diversamente dipendentemente dalla modalità di voto.

Si è scelto di implementare il passaggio di informazioni tra scene dell'interfaccia grafica tramite messaggi. In particolare, prima di caricare la nuova scena è possibile invocare metodi setter per impostare gli oggetti che si ritiene siano necessari anche alla scena di arrivo. Questo approccio non è ottimale quanto a accoppiamento, in quanto oggetti del modello vengono mantenuti e manipolati in oggetti che gestiscono l'interfaccia grafica. Tuttavia, si è prediletta questa modalità al mantenimento di tutte le informazioni comuni tra scene nello stage genitore. Si è preferito, infatti, avere un controllo meticoloso sulle informazioni accessibili da ciascuna scena, onde ridurre la possibilità di errore e di problemi di accessi non consentiti.

Da notare che con la classe *SchedaElettorale* si intende un template di scheda elettorale per una data consultazione. Per questo motivo gli oggetti di tipo *Preferenza* sono stati pensati separatamente rispetto a *SchedaElettorale*. Visto che nella classe *Preferenza* la mappatura ha come chiave una voce, il cui identificativo la lega alla scheda elettorale, si è pensato essere ridondante inserire una collezione di preferenze all'interno di *SchedaElettorale*. Visto anche l'utilizzo one time di oggetti tipo *Preferenza* - per inserimento o recupero dal database - è parso opportuno riportare in *SchedaElettorale* solamente il risultato della consultazione, cioè il cumulativo delle preferenze espresse durante il periodo di voto.

4.7 Gestione dati persistenti

La struttura del database è descritta nella figura 4.32, che contiene lo schema ER dello stesso. Com'è possibile osservare le entità del database ricalcano le classi del modello del programma, oppure permettono di rappresentare collezioni di oggetti, qualora nelle associazioni tra classi fossero previste collezioni (vedi diagramma delle classi 3.2).

Per gestire il polimorfismo dell'interfaccia *Voce*, è stata prevista l'entità *informazione scheda*, che funge come una sorta di dispatcher tra le possibili classi che implementano *Voce*, quali *Quesito*, *Candidato* e *Partito*. In linea con il modello, in cui gli oggetti di classi che implementano l'interfaccia *Voce* compongono sia *InformazioneScheda* che *Preferenza*, nel database le preferenze sono mantenute solo in relazione a voci che compongono l'informazione di una data scheda.

Per le tabelle *cae* e *utente* sono previsti trigger all'atto dell'inserimento e della modifica affinché nel campo *password* sia sempre presente l'hash della stringa inserita dall'utente. In altri termini il

database mantiene sempre e solo `sha1(password)`, nel rispetto del requisito *SN1* (2.2.2).

4.8 Descrizione interfaccia grafica

La navigation map nella figura 4.33 indica come e tra quali scene è possibile navigare. Le scene sono denominate con i nomi dei file in *resources/views*.

Ciascun passaggio tra scene è dettato dal verificarsi di determinati eventi, in genere la pressione di bottoni appositi. In alcuni casi, come evidenziato dai commenti sulle navigabilità, i passaggi tra scene sono riservati a determinati tipi di utente, con riferimento agli attori del diagramma dei casi d'uso (3.1).

Segue una breve descrizione della particolarità di alcune scene.

areaCAE Nel caso in cui l'utente è di tipo CAE, il sistema dà la possibilità di eseguire i procedimenti per cui ha i diritti. Il CAE configuratore e scrutatore vedrà rispettivamente una scena tipo 4.34 oppure 4.35.

Autenticazione

accessoSPIDCIE Visto che non è stata implementata l'interazione con l'accesso tramite SPID e CIE (vedi 5.1), è stata prevista un'autenticazione utente tipo username - password. Si assume che le credenziali degli utenti registrati siano mantenuti nel database. La scena *areaSPIDCIE* gestisce questa autenticazione. Vengono segnalati all'utente email inserite con formato non valido (figura 4.36), e l'inserimento di credenziali errate, incluso il caso in cui l'email non è registrata (figura 4.37).

All'utente sono permessi un massimo di tre tentativi di accesso (non inclusi gli inserimenti di email con formato non valido); dopo il terzo inserimento di credenziali errate, l'utente viene reindirizzato alla pagina di benvenuto.

autenticazioneCAE In questa scena, come possibile vedere nella navigation map, può arrivare solamente un utente di tipo CAE. Anche in questo caso sono ammessi al massimo 3 tentativi per l'inserimento della password; dopodiché viene mostrata la schermata di benvenuto.

Procedimento di voto

inserimentoSchedeElettorali Questa scena permette al CAE di inserire schede tradizionali. In particolare può registrare che un elettore ha espresso una preferenza per una data consultazione, oppure può inserire la preferenza per una delle consultazioni in corso. In entrambi casi il CAE può accedere solamente alle consultazioni per cui ha i permessi.

Anche in questo caso viene effettuato un controllo sulla validità del formato dell'email inserito. In aggiunta, si verifica che l'email appartenga ad un elettore registrato nel sistema; in caso contrario la scena si presenta come in figura 4.38. Successivamente al CAE viene data la possibilità di scegliere tra le consultazioni previste nel periodo di voto corrente per l'elettore inserito. Nel caso in cui l'intersezione tra le consultazioni attive per l'elettore e quelle per cui il CAE ha permessi è vuoto l'interfaccia notifica l'utente come mostrato in figura 4.39.

areaElettore Da qui l'elettore può scegliere la consultazione per cui votare. La scena 4.40 viene mostrata nel caso in cui non risultano disponibili consultazioni per l'elettore. Questo può avvenire nel caso in cui:

- Non sono attive sessioni per l'elettore (con riferimento al luogo di residenza) nel periodo di voto corrente
- Le schede in una sessione attiva hanno un limite di età che supera l'età dell'elettore

Procedimento di configurazione

areaCreaSessione Questa scena permette di proseguire nella creazione di una nuova sessione a condizione che siano valorizzati tutti i campi mostrati. È possibile inserire solamente date di inizio sessione a partire dal giorno corrente, e la data di fine è selezionabile solo a partire dal giorno selezionato per l'inizio della sessione. Nel complesso la scena si presenta come mostrato nella figura 4.41

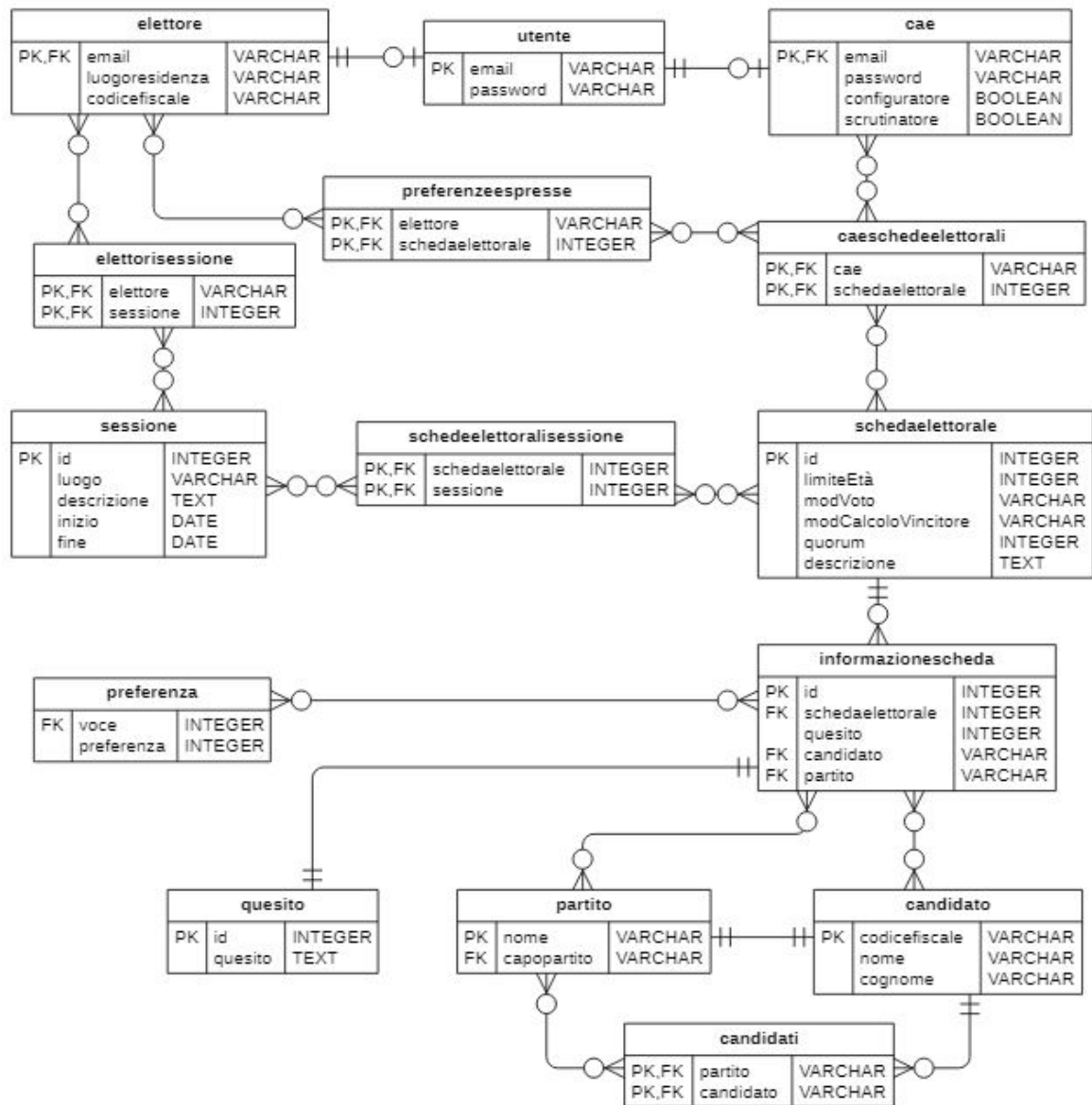


Figura 4.32: Schema ER del database

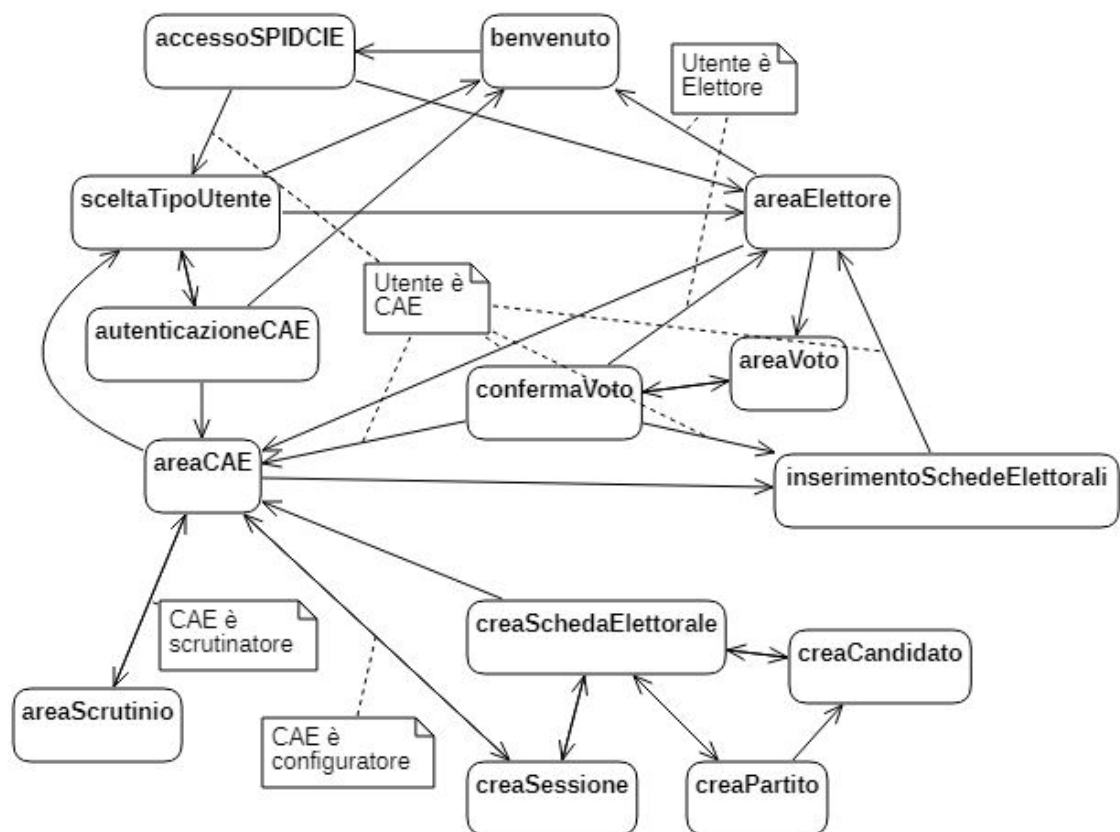


Figura 4.33: Navigation map dell'interfaccia grafica

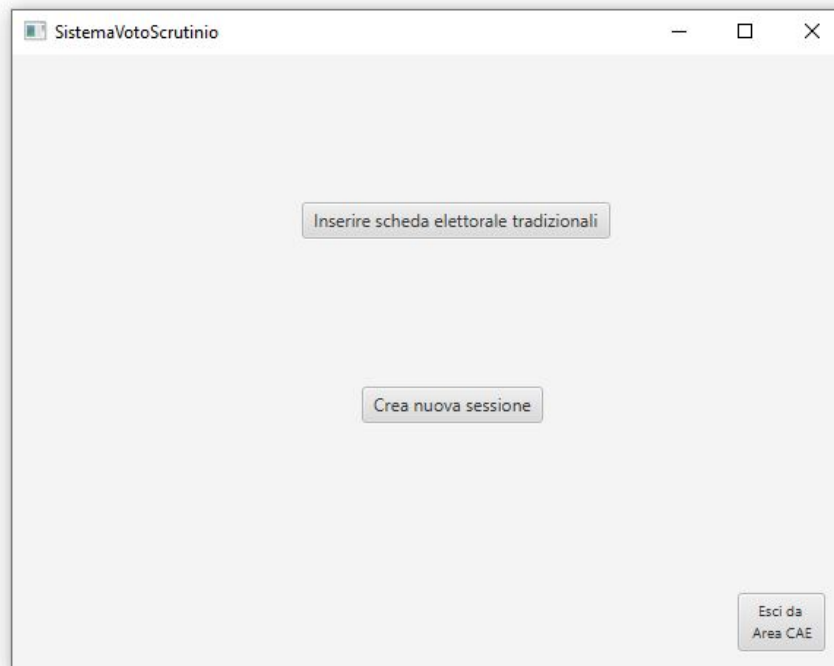


Figura 4.34: Area di un CAE configuratore

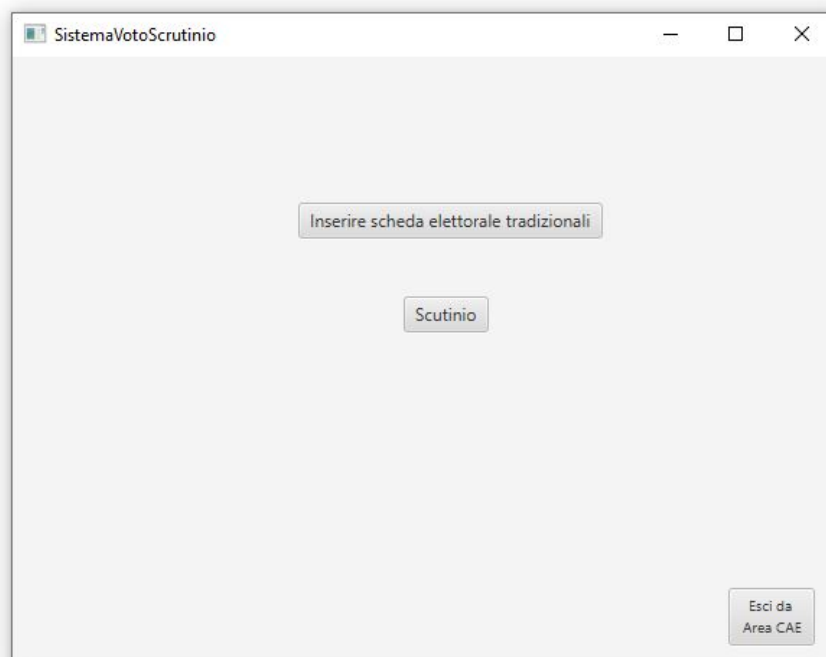


Figura 4.35: Area di un CAE scrutatore

areaCreaSchedaElettorale Anche in questo caso la consultazione associata alla scheda elettorale impostata può essere creata a condizione che siano valorizzati tutti i campi della scena [4.42](#).

Le scelte per la modalità di calcolo del vincitore e per l'informazione della scheda variano in funzione della modalità di voto selezionata. Nelle figure [4.43](#), [4.44](#) e [4.45](#) vengono mostrati alcuni esempi della variabilità delle possibili scelte.

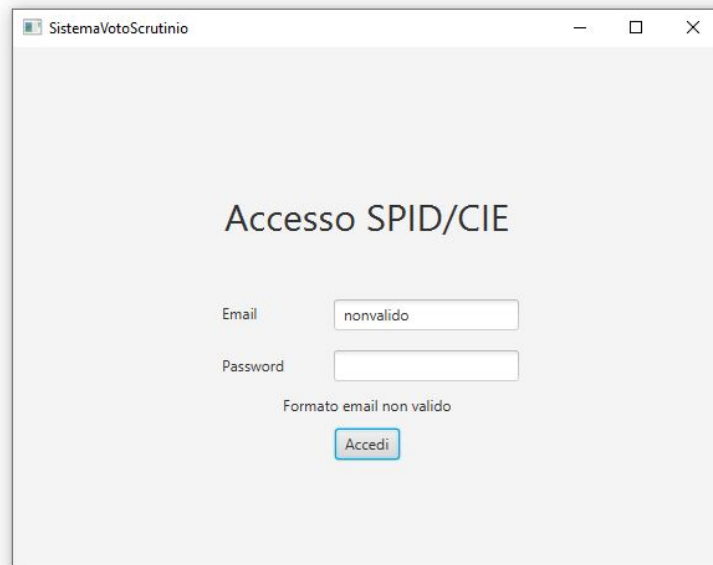


Figura 4.36: Autenticazione utente - email in formato non valido

areaCreaPartito Viene data la possibilità di creare un nuovo partito. È possibile inserire nel partito solamente candidati che non appartengono ad altri partiti esistenti, e il capo partito è scegliibile solamente tra i candidati selezionati.

areaCreaCandidato In questa scena è possibile impostare un nuovo candidato da inserire nel sistema. Viene effettuato un controllo sulla validità del codice fiscale.

Procedimento di scrutinio

areaScrutinio In questa area vengono mostrati al CAE scrutatore un sottoinsieme delle consultazioni per cui è possibile effettuare lo scrutinio.

Una consultazione viene mostrata a condizione che:

- Lo scrutatore abbia permessi sulla consultazione in questione
- Non è nullo il numero di elettori associati alla consultazione
- È terminato il periodo di voto per ciascuna sessione in cui è contenuta la consultazione

4.9 Testing

Seguono le descrizioni dei test eseguiti per il pacchetto *model* e per l'interfaccia grafica. Non sono state testate le classi del pacchetto *database*, dal momento che raccolgono istruzioni propedeutiche ad interrogare il database tramite comandi SQL, assunti come corretti nella loro interazione con i dati persistenti. Per questo motivo si estende il concetto di metodi getter anche a quei metodi che restituiscono risultati ottenuti interrogando il database.

4.9.1 Classi di *model*

In fase di testing sono state fatte assunzioni che hanno portato ad una selezione dei metodi per cui implementare test driver. Seguono le motivazioni per i test non effettuati.

- I costruttori sono assunti come corretti poiché semplicemente assegnano agli attributi delle classi i valori passati in input, oppure inizializzano gli attributi in modo da rispettare l'invariante della classe, per come espresso dai vincoli descritti nella sezione 4.2.1, assunti come rispettati.

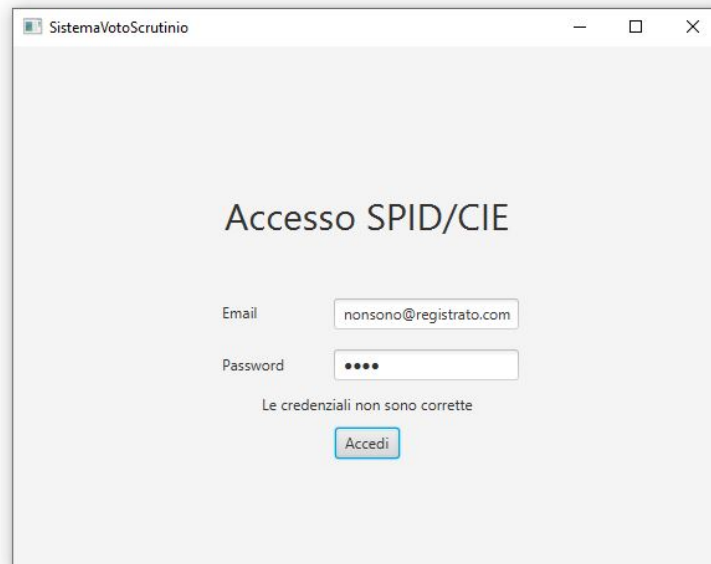


Figura 4.37: Autenticazione utente - email non registrata

- Non sono stati eseguiti test su metodi getter semplici, cioè che non comportano un filtraggio dei valori restituiti, poiché restituiscono il valore di un attributo dell'oggetto, oppure invocano metodi del pacchetto *database*, che, come spiegato al punto 4.9, sono assunti come corretti. Dal punto di vista di copertura del codice, i test svolti sugli altri metodi sono stati sufficienti per eseguire anche i metodi getter citati.
- È stata fatta la scelta di non testare metodi basilari, quali *toString*, *equals*, *hashCode* e *iterator*, dato che seguono un template standard di implementazione.

Gli altri metodi sono stati testati con il criterio di copertura MCDC. A seguire, per ciascun metodo di ciascuna classe, vengono riportati in forma schematica i rami di decisione del programma considerati per coprire il codice secondo il criterio scelto. I test driver sono stati denominati seguendo la convenzione `[nome_metodo_testato][indicazione_valore_verità_condizioni]`.

Tutti i test eseguiti hanno avuto esito negativo, cioè non hanno causato malfunzionamenti.

Cae

```
public boolean scrutinio(SchedaElettorale s)

    if(this.scrutinatore){
        if(result){}
    }

public void creaSchedaElettorale(SchedaElettorale s)

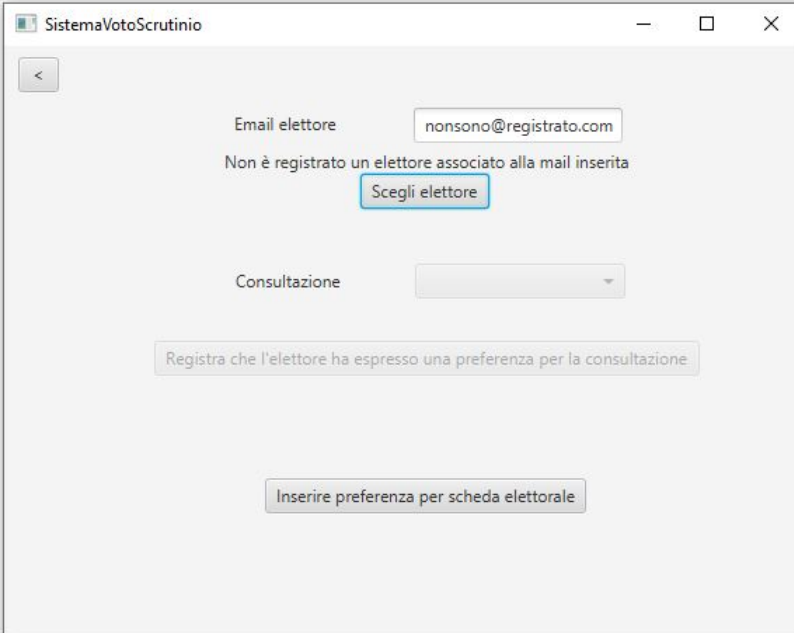
    if(this.configuratore)

private boolean configuratorePerScheda(SchedaElettorale s)

    if(!this.configuratore){
        if(!getSchedeElettorali().contains(s)){
        }
    }

public Set<SchedaElettorale> getSchedeElettoraliConElettori()

    if(s.numeroElettoriTotali()>0){}
```



SistemaVotoScrutinio

<

Email elettore

Non è registrato un elettore associato alla mail inserita

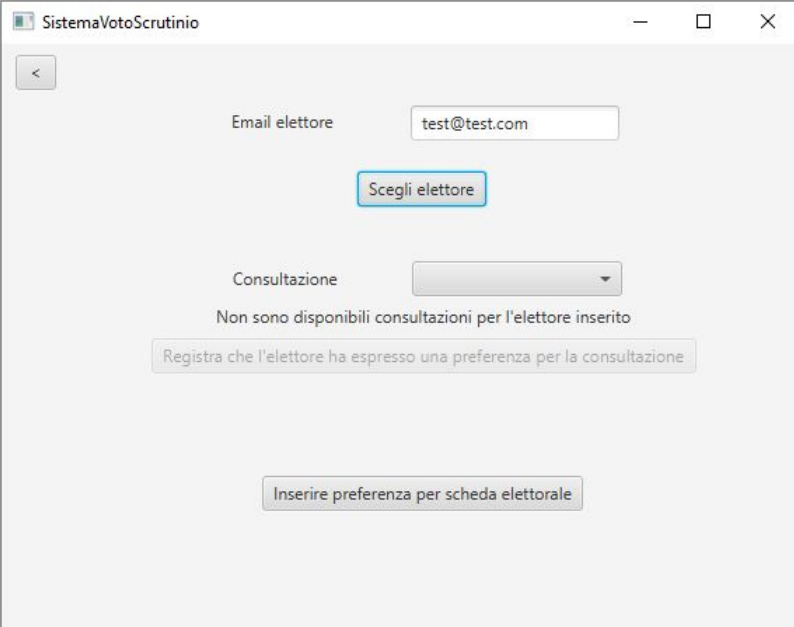
Scegli elettore

Consultazione

Registra che l'elettore ha espresso una preferenza per la consultazione

Inserire preferenza per scheda elettorale

Figura 4.38: Inserimento schede tradizionali - elettore non registrato



SistemaVotoScrutinio

<

Email elettore

Scegli elettore

Consultazione

Non sono disponibili consultazioni per l'elettore inserito

Registra che l'elettore ha espresso una preferenza per la consultazione

Inserire preferenza per scheda elettorale

Figura 4.39: Inserimento schede tradizionali - nessuna consultazione disponibile

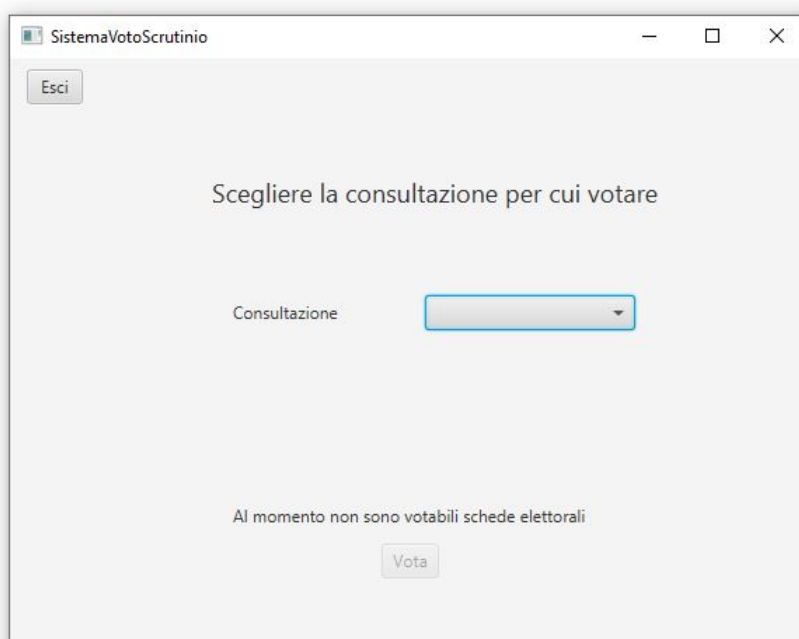


Figura 4.40: Area di voto - nessuna consultazione disponibile nel periodo di voto corrente

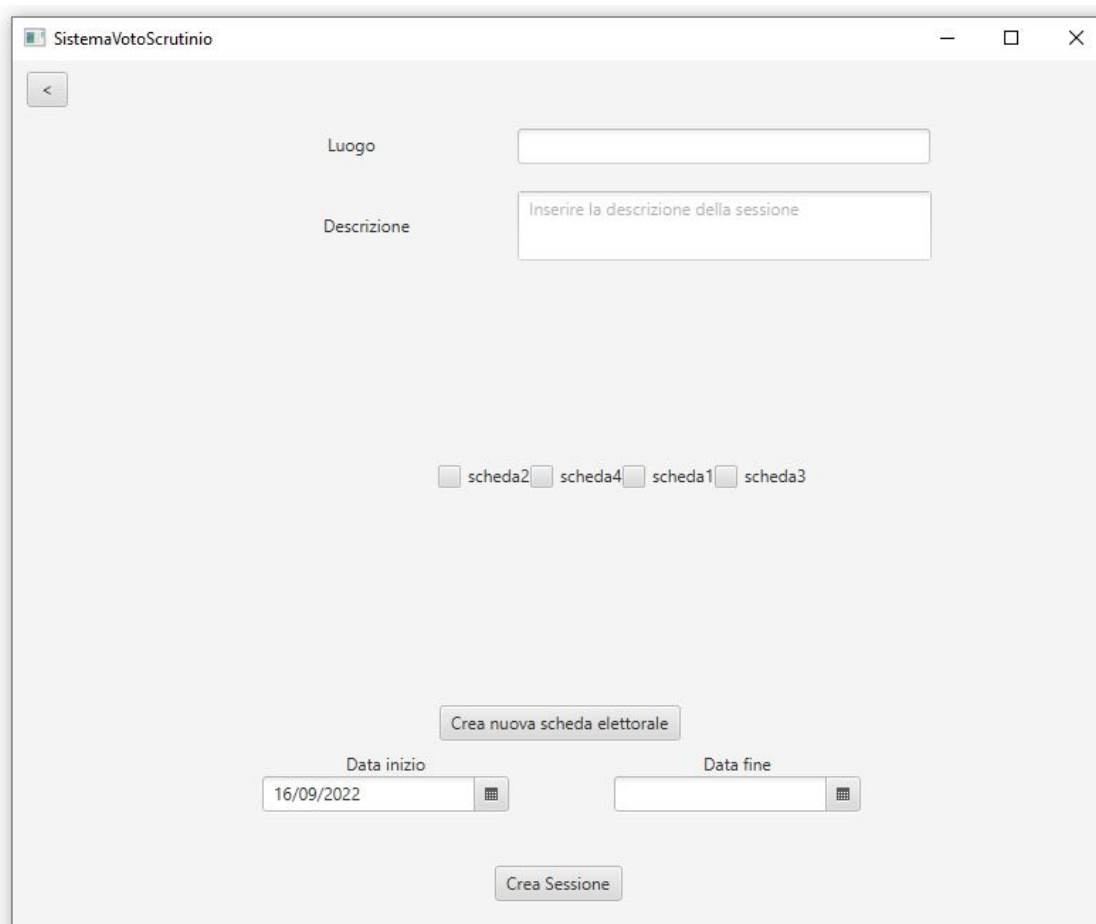


Figura 4.41: Area di configurazione per la creazione di una nuova sessione di voto

The screenshot shows a window titled "SistemaVotoScrutinio" with standard Windows window controls (minimize, maximize, close). Inside the window, there is a back button (left arrow) in the top-left corner. The main area contains five configuration fields:

- Descrizione:** A text input field.
- Limite età:** A numeric input field with the value "18" and up/down arrow buttons.
- Modalità di voto:** A dropdown menu.
- Modalità di scrutinio:** A dropdown menu.
- Percentuale quorum:** A numeric input field with the value "0" and up/down arrow buttons.

At the bottom center of the window is a button labeled "Crea Scheda".

Figura 4.42: Area di configurazione per la creazione di una nuova scheda elettorale

SistemaVotoScrutinio

<

Descrizione

Limite età 18

Modalità di voto Categorico con preferenze

Modalità di scrutinio

Percentuale quorum 0

Scegliere i partiti da inserire nella scheda

☐ Partito2 - capo partito: Candidato Secondo ☐ Partito1 - capo partito: Candidato Primo

Crea partito

Crea Scheda

Figura 4.43: Area di configurazione per la creazione di una nuova scheda elettorale - modalità di voto categorico con preferenze

The screenshot shows a software window titled "SistemaVotoScrutinio". Inside, there is a configuration form for creating an electoral schedule. The form includes the following elements:

- A back button (left arrow) in the top-left corner.
- A text label "Descrizione" followed by an empty text input field.
- A text label "Limite età" followed by a numeric input field containing the value "18".
- A text label "Modalità di voto" followed by a dropdown menu currently showing "Ordinale".
- A text label "Modalità di scrutinio" followed by a disabled dropdown menu.
- A text label "Percentuale quorum" followed by a numeric input field containing the value "0".
- A text label "Scegliere la categoria tra cui andrà effettuata la scelta" followed by two radio buttons: "Partiti" (selected) and "Candidati".
- A "Crea Scheda" button at the bottom center.

Figura 4.44: Area di configurazione per la creazione di una nuova scheda elettorale - modalità di voto ordinale

The screenshot shows a software window titled "SistemaVotoScrutinio". Inside the window, there is a configuration form for creating a new electoral schedule. The form includes the following elements:

- A back button (left arrow) in the top-left corner.
- A "Descrizione" label followed by a text input field.
- A "Limite età" label followed by a numeric input field containing the value "18".
- A "Modalità di voto" label followed by a dropdown menu currently set to "Referendum".
- A "Modalità di scrutinio" label followed by a disabled dropdown menu.
- A "Percentuale quorum" label followed by a numeric input field containing the value "0".
- A large text area with the placeholder text "Inserisci il testo del quesito".
- A "Crea Scheda" button at the bottom center.

Figura 4.45: Area di configurazione per la creazione di una nuova scheda elettorale - modalità di voto referendum

Candidato

```
public void crea()
    if(!cDAO.esiste(this)){
public void elimina()
    if(cDAO.esiste(this)){}
```

Elettore

```
public void crea()
    if(!eDAO.esiste(this)){
public void elimina()
    if(eDAO.esiste(this)){
public Set<Sessione> getSessioniInCorso()
    sessioni.removeIf(c -> (c.getDataInizio().after(date)
        || c.getDataFine().before(date)));
public Set<SchedaElettorale> getSchedeElettorali()
    if(this.autorizzato(s.getLimiteEta())){
public Set<SchedaElettorale> getSchedeElettoraliSenzaPreferenza()
    if(this.autorizzato(s.getLimiteEta()) && !eDAO.preferenzaEspressa(this, s)) {}
```

ModCalcoloVincitore

```
public Voce calcoloVincitore(Risultato r)
    MAGGIORANZA
    if(listsize>1
        && list.get(listsize-2).getValue() == list.get(listsize-1).getValue()){
    MAGGIORANZAASS
    if(!Objects.isNull(vincitore)
        && r.getRisultato().get(vincitore) >= r.getNumeroVotanti()/2 + 1){}
```

Partito

```
public void crea()
    if(!pDAO.esiste(this)){
public void elimina()
    if(pDAO.esiste(this)){}
```

Quesito

```
public void crea()
    if(!qDAO.esiste(this)){
public void elimina()
    if(qDAO.esiste(this)){}
```

Risultato

```
public void calcolaVincitore()
    if(!risultato.isEmpty()) {}

public void add(Voce v, int ris)
    if(!risultato.isEmpty()){}
```

SchedaElettorale

```
public void crea()
    if(!sDAO.esiste(this)){}
```

```
public void elimina()
    if(sDAO.esiste(this)){}
```

```
public boolean scrutinio()
    if(c.getDataFine().after(date)) {
    }else if(Objects.isNull(risultato)) {
        if(risultato.getNumeroVotanti() < this.getQuorum()) {}
    }
```

Utente

```
public void crea()
    if(!uDAO.esiste(this)){}
```

```
public void elimina()
    if(uDAO.esiste(this)){}
```

4.9.2 Interfaccia grafica

Sono stati svolti numerosi test sull'interfaccia grafica. Si riportano alcuni degli aspetti che sono stati controllati.

- Il corretto susseguirsi delle scene con la pressione degli appositi bottoni, nel rispetto della navigabilità delle scene (vedi [4.33](#))
- Nel caso di cambiamento di scena, la corretta impostazione degli oggetti nella scena di arrivo, in linea col diagramma [3.2.3](#)
- Che le opzioni mostrate dipendessero dai permessi dell'utente
- I campi mostrati permettono la valorizzazione soltanto nelle opportune condizioni
- Notifica all'utente in caso di non valorizzazione di campi, o di inserimento di stringhe vuote o contenenti soltanto spazi
- ... le peculiarità di ciascuna scena descritta nella sezione [4.8](#)

4.9.3 Database

È stato controllato che l'inserimento di dati nel procedimento di configurazione e di voto avvenga correttamente, e che per tutti i casi d'uso del sistema vengano prelevate le informazioni opportune dal database.

Capitolo 5

Considerazioni finali

5.1 Discrepanze progettazione - implementazione

Come è possibile notare non tutti i requisiti iniziali sono stati rispettati nell'implementazione del sistema. Questo è in gran parte dovuto alla mancata possibilità di utilizzare strumenti previsti per la gestione di autenticazione e comunicazione con reali elettori cittadini italiani.

A seguire sono riportati i requisiti non rispettati con relativa motivazione.

FU2.2.3, FS5.3, S4 Non è stata implementata la gestione dell'invio mail, dunque neppure quella per comunicare la corretta registrazione della preferenza espressa dall'elettore, né quella tramite PEC agli OPRU alla conclusione del procedimento di scrutinio.

SE2 Dato che non è prevista la comunicazione dei risultati agli OPRU, i risultati vengono mostrati a schermo al CAE che ha eseguito lo scrutinio.

FS1 Il sistema non implementa l'autenticazione tramite SPID e CIE, e sopperisce prevedendo una autenticazione tipo username - password. Dal momento che sostituisce SPID e CIE non sono previste le modalità di modifica e inserimento utenti nel sistema: si assume, difatti, che i cittadini aventi diritto al voto, che possono dunque usufruire dei servizi del sistema, siano già inseriti in un database nazionale.

FS3.1 Come per le altre informazioni riguardanti l'elettore, la residenza è salvata nel database del sistema, quindi non viene recuperata dalle informazioni associate alla sua identità digitale.

Q1 Al momento il sistema non è facilmente accessibile a utenti non vedenti o con impedimenti gravi.

Q2.1 Non è stata prevista la gestione della ripresa del sistema in caso di malfunzionamenti. È assicurato, però, che a meno di esplicita conferma non vengono registrate configurazioni o espressioni di preferenze per consultazioni.

5.2 Sviluppi futuri

Un insieme di potenziali aggiunte future mira a sopperire le lacune che causano la non conformità con i requisiti elencati al punto precedente. In particolare si può pensare di:

- Implementare la gestione dell'invio mail; a tale scopo sono previste le signature dei metodi *Cae.comunicaRisultato* e *Elettore.comunicaPreferenzaEspressa*, che vengono invocati nei punti opportuni per rendere soddisfatti i requisiti. Nel caso della comunicazione dei risultati degli scrutini è da eliminare la stampa a schermo di quest'ultimi nell'interfaccia grafica nell'area di scrutinio.
- Implementare l'accesso tramite SPID e CIE, permettendo di ricavare in automatico le informazioni riguardanti gli elettori. In particolare per la residenza, nella configurazione di nuove consultazioni, ha senso prevedere la scelta di un luogo (comune, regione, nazione) da un elenco gestito a livello nazionale.
- Aggiungere la possibilità di ascoltare il testo mostrato dall'interfaccia grafica.

- Prevedere un modo per segnalare malfunzionamenti all'utente, e gestire la successiva ripresa, anch'essa in modo user friendly.

Ulteriori proposte sono:

- Gestire il blocco dell'utente nel caso in cui vengano effettuati troppi tentativi di inserimento credenziali.
- Implementare il sistema con un'architettura client-server, per permettere una esplicita gestione di un gran numero di utenti, considerato l'ambito di applicazione del sistema.

5.3 Note per l'installazione e l'utilizzo

Per l'avviamento del programma:

- main class: `controllers.Main`
- VM arguments: `--module-path "[path_to_javafx]" --add-modules javafx.controls,javafx.fxml`
- per l'eventuale accesso all'istanza locale del database: `username root - password root`

Altre dipendenze necessarie, quali il connettore MySQL e JUnit, sono gestite tramite il pom del progetto Maven.

5.3.1 Contenuto del database

Nella cartella *database* è presente il dump di un'istanza esempio del database. Il sistema non prevede l'inserimento e la modifica di credenziali di utenti, sia elettori che CAE (si assume che questi dati vengano automaticamente inseriti con l'accesso tramite SPID), dunque queste operazioni vanno effettuate direttamente sul database, oppure sfruttando i metodi *crea* ed *elimina* nelle classi in *database*.

Le tabelle nella pagina seguente elencano i dati già presenti nel database.

Tabella 5.1: Utenti

<i>email</i>	<i>password</i>
prova@prova.com	prova
test@test.com	test

Tabella 5.2: Elettori

<i>email</i>	<i>luogo residenza</i>
prova@prova.com	Milano
test@test.com	Roma

Tabella 5.3: CAE

<i>email</i>	<i>password</i>	<i>configuratore</i>	<i>scrutinatore</i>
prova@prova.com	1234	✓	✓

Tabella 5.4: Candidati

	<i>nome</i>	<i>cognome</i>	<i>codice fiscale</i>
Candidato	Uno	AAAAAA	A00A00A000A
Candidato	Due	BBBBBB	B11B11B111B

Tabella 5.5: Partiti

<i>nome</i>	<i>capo partito</i>
Partito1	Candidato Uno

Tabella 5.6: Consultazioni

<i>descrizione</i>	<i>età</i>	<i>modalità voto</i>	<i>modalità calcolo vincitore</i>	<i>livello quorum (%)</i>	<i>voci</i>
Consultazione Ordinale	18	Ordinale	Maggioranza Ass.	0	Candidato Uno Candidato Due
Consultazione Categorico	25	Categorico	Maggioranza	0	Partito1
Consultazione Categorico con preferenze	18	Categorico con preferenze	Maggioranza	0	Partito1
Consultazione Referendum	18	Referendum	-	50	Referendum

Tabella 5.7: Sessioni

<i>descrizione</i>	<i>luogo</i>	<i>inizio</i>	<i>fine</i>	<i>consultazioni</i>
Sessione1	Roma	21/09/2022	30/09/2022	Consultazione Ordinale Consultazione Categorico con preferenze
Sessione2	Milano	21/09/2022	31/12/2022	Consultazione Categorico Consultazione Ordinale Consultazione Referendum