

DOCUMENTAZIONE SISTEMA DI VOTO

Lorenzo Gardelli, matricola 907502

Per il corso di Ingegneria del Software, AA 2021/2022

Sommario

Introduzione	3
Assignment:.....	3
Indicazioni di lettura:.....	4
Note per l'insegnate:.....	4
Sistema richiesto + specifiche aggiunte	5
1 Fase di Analisi/Specifica Requisiti	6
1.1 Obiettivi:	6
1.2 Destinatari:.....	6
1.3 Scopo del sistema:.....	6
1.4 Glossario:	7
1.5 Requisiti funzionali:	8
1.5.1 Requisiti funzionali utente:.....	8
1.6 Requisiti non funzionali:	9
1.6.1 Requisiti esterni:.....	10
1.6.2 Requisiti d sicurezza:	11
1.7 User stories:	12
2 Progettazione del sistema	14
2.1 Use Case	14
2.2 Scenari.....	16
2.2.1 Scenario 1 (Utente generico login)	16
2.2.2 Scenario 2 (Utente ipovedente).....	17
2.2.3 Scenario 3 (Guasto del dispositivo di voto)	17
2.2.4 Scenario 4 (Pubblicazione risultati).....	18
2.3 Diagramma delle classi	20

High Cohesion:	21
Low Coupling:	21
2.4 Diagrammi di sequenza	22
Sequenza scenario 1:	22
Sequenza scenario 2:	23
Sequenza scenario 3:	24
Sequenza scenario 4:	25
2.5 Diagrammi di attività	26
Activity 1	26
Activity 2	27
2.6 Diagrammi di stato	28
2.7 Diagramma di Deployment.....	32
3 Implementazione del sistema	33
3.1 Design Pattern.....	33
3.1.1 MVC.....	33
3.1.2 Singleton	34
3.1.3 DAO	36
3.2 Vincoli	37
3.3 Testing/Controllo parametri input.....	38
3.4 Gestione dati persistenti	40
3.5 Navigation map	43
3.6 Schermate di esempio	44
3.6.1 Esempi controllo input.....	48

Introduzione

Relazione del progetto “Sistema di voto e di scrutinio elettronico” valida per il corso di Ingegneria del Software, AA 2021/2022.

Ogni specifica aggiunta o interpretazione data al tema di progetto (volutamente incompleto) è stata chiaramente indicata nel seguente documento, correlata di spiegazione riguardo al perché è stata aggiunta e quali modifiche ha comportato.

Il link di riferimento al progetto completo è il seguente:

https://github.com/LorenzoGardelli/INGSW_SistemaDiVoto.git

Nella pagina sono presenti copia di questo documento e varie cartelle di supporto.

Assignment:

La realizzazione della relazione si è basata, specie nelle fasi iniziali, sugli assignment richiesti e inviati durante il corso, valutati tutti in maniera positiva. Come richiesto dalla consegna, ecco l’elenco delle modifiche aggiunte agli assignment rispetto a quando vennero caricati per la valutazione:

- Assignment 1: il documento RASD prodotto è stato aggiornato con nuove specifiche più dettagliate, anche a seguito di alcune ambiguità riscontrate solo nelle fasi successive di progettazione. Il documento così aggiornato è stato poi inserito nella relazione e copre la quasi totalità del [capitolo 1](#) (“Fase di analisi/Specifica requisiti”).
- Assignment 2: il progetto e la documentazione caricata su GitHub è stata fortemente modificata e aggiornata. Il nome del repository è stato aggiornato.
- Assignment 3: La classe elettore è stata ampliata, i vincoli JML presi come esempio per i nuovi vincoli OCL. È stata presa come classe di esempio per alcune osservazioni e per alcuni vincoli/test.
- Assignment 4: Aggiunti nuovi esempi di diagrammi e scenari ritenuti più rappresentativi, e quelli precedentemente realizzati sono stati aggiornati alle nuove specifiche e all’introduzione di alcuni Design Pattern.
- Assignment 5: La finestra di login è stata ridisegnata, sono poi state aggiunte le altre schermate della GUI cercando di seguirne lo stesso stile grafico. È stata mantenuta la logica del controllo degli input.

La valutazione di tutti e cinque gli assignment era risultata POSITIVA.

Indicazioni di lettura:

Uno dei principi chiave sul quale si basa la relazione è la tracciabilità. Sono presenti continui riferimenti tra i vari capitoli, a cominciare dai requisiti la cui implementazione influenza ogni parte dell'analisi e dell'implementazione. Gli stessi diagrammi UML permettono di rappresentare aspetti e viste diverse dello stesso sistema e sono perciò fortemente collegati tra loro.

Per questo motivo durante la lettura quando si incontra un riferimento a un capitolo diverso è stato aggiunto un link rapido al riferimento in questione.

I diagrammi e le schermate di esempio sono stati ridimensionati per essere inseriti nel documento, e in alcuni casi la leggibilità è diminuita. All'interno del progetto è presente la cartella "Immagini" contenente gli screenshot originali con qualità di dettagli maggiore. Durante la lettura si consiglia di tenere aperti a lato i diagrammi originali per avere un confronto immediato, soprattutto nel caso degli Use Case e Classi.

Note per l'insegnate:

Si ha la necessità di comunicare che nel mio piano di studi non è presente il corso di Programmazione 2 tenuto dal Prof. Santini. Le conoscenze di programmazione vera e propria derivano dal corso di Programmazione 1 (linguaggio C), Algoritmi e strutture dati e Statistica (Python). In nessuno di questi corsi è stato studiato Java né è stato indicato come creare un applicativo completo e funzionante.

L'implementazione a livello di codice risulterà incompleta. Per la parte di realizzazione vera e propria del sistema ho perciò dato massima priorità all'implementazione dell'interfaccia grafica rispetto al codice che vi è dietro, in modo da poterne mostrare comunque un esempio di funzionamento attraverso la grafica delle varie finestre di utilizzo. A livello di codice, ho preso come esempio per diversi punti della relazione la classe Elettore, già implementata per via dell'assignment 3, mentre per l'applicazione dei Design Pattern la classe target d'esempio è ResponsabileGenerale. Purtroppo, non ho modo nemmeno di aggiungere Programmazione 2 agli insegnamenti a scelta.

Sistema richiesto + specifiche aggiunte

Il progetto richiesto dal corso per l'AA 2021/2022 riguarda la progettazione di un sistema di voto e di scrutinio elettronico, generico e in grado di supportare diverse modalità di voto e di definizione dei vincitori.

Si ipotizza il seguente funzionamento: nelle varie sedi di voto sono presenti postazioni fisse dove i cittadini possono esprimere il loro voto. In queste postazioni sarà presente un computer (da qui in avanti chiamato Totem) dotato di schermo touch, casse e connessione a internet stabile. Su tali dispositivi sarà installato il software "client" per la votazione.

Prima delle votazioni, ogni singola postazione verrà configurata da tecnici appositi che collegano il software sul computer al server centrale del sistema di voto (maggiori specifiche relative al server sono presenti nei requisiti successivi).

Al termine delle votazioni, il sistema pubblicherà i risultati. L'intero sistema deve essere a prova di attacchi di sicurezza per scongiurare eventuali brogli elettorali (di nuovo, vedere le specifiche successive).

Rispetto alle specifiche iniziali, oltre agli utenti Elettore e Tecnico Gestore del sistema, si è scelto di aggiungere una terza figura: Responsabile Generale.

Questo perché non sembra verosimile che, durante un'elezione a livello statale, tra tutte le migliaia di Tecnici di sistema ne venga individuato solo uno per inserire i dati di votazione. Ai tecnici andrà il compito di configurare, settare, avviare e provvedere alla manutenzione dei dispositivi fisici. Sarà invece compito del Responsabile Generale provvedere all'inserimento dei parametri per una nuova votazione (tipologia, quorum, lista partiti ed esponenti) possibilmente giorni prima della data di voto e senza necessità di autenticarsi e usare un dispositivo di voto, ma agendo direttamente in locale nel sistema generale.

Il Responsabile Generale va considerato come il tramite tra il sistema di voto e il Ministero degli Interni (che fornisce i dati dei partiti registrati) o dei comuni per i quali si vota. Sarà poi compito di tale Responsabile al termine delle votazioni, verificare la mancata presenza di violazioni di sicurezza e confermare quindi la pubblicazione dei risultati.

1 Fase di Analisi/Specifica Requisiti

1.1 Obiettivi:

L'obiettivo di questa fase iniziale di specifica/ analisi requisiti è quello di creare un documento che riporti in maniera dettagliata e non ambigua le funzionalità e le proprietà che l'applicazione Software dovrà avere.

Il documento RASD (Requirements Analysis and Specification Document) dovrà essere un punto d'incontro tra il cliente, il quale potrà così verificare se il progetto teorizzato ha i requisiti da lui richiesti, e il progettista, che utilizzerà tale documento come punto di partenza per lo sviluppo del prodotto.

1.2 Destinatari:

Si ipotizza che la visione e l'utilizzo di tale documento sia rivolto a queste figure:

- **Agenzia governativa**, ovvero il cliente che commissiona la realizzazione del prodotto, che necessita del documento per accertarsi che le specifiche richieste per il software siano corrette, complete, e a norma di legge.
- **Project manager e programmatore**, coloro che devono effettivamente realizzare il sistema di voto, che utilizzano il documento come riferimento su cui basare la progettazione, realizzazione e testing del software.
- **Cittadino**, non direttamente coinvolto nel progetto. Essendo il software destinato a gestire il sistema elettorale, per motivi di trasparenza è possibile che il seguente documento venga reso pubblico per consentirne una libera consultazione.

1.3 Scopo del sistema:

Scopo del progetto è di realizzare un software per la gestione elettronica del sistema di voto e scrutinio.

Attraverso questo sistema sarà possibile impostare delle votazioni o dei referendum e permettere ai cittadini di esprimere il loro voto.

Il sistema dovrà essere utilizzabile sia per elezioni locali (comunali) che per elezioni generali statali, e dovrà essere particolarmente attento alla sicurezza di fronte a possibili attacchi e alla gestione dei dati e delle informazioni personali e sensibili.

1.4 Glossario:

Referendum: votazione basata su una singola domanda, alla quale i cittadini devono rispondere se sono favorevoli o contrari.

Quorum: indica il numero minimo di voti necessari perché il referendum possa essere considerato valido. Normalmente corrisponde alla maggioranza degli aventi diritto di voto.

Voto ordinale: Votazione nella quale è richiesto che l'elettore ordini i candidati o partiti in base alle proprie preferenze

Voto categorico: Votazione nella quale è richiesto che l'elettore indichi una preferenza per un candidato o partito.

Partito: In base alla Costituzione, associazione libera di cittadini che detengono il diritto di amministrare democraticamente la vita politica.

SPID: Sistema Pubblico di Identità Digitale che permette ai cittadini di identificarsi univocamente per accedere a servizi anche online.

Autenticazione: Verifica della propria identità da parte del sistema.

Exit Poll: Risultati parziali delle votazioni basati su speculazioni o analisi parziale delle schede elettorali. La loro pubblicazione può influenzare l'opinione pubblica e l'andamento delle elezioni.

Expected User: Tipologie di utenti che ci si aspetta debbano utilizzare/interagire col software.

Voto Segreto: Tutela che assicura all'Elettore che la preferenza espressa non sia in nessun modo riconducibile a lui. La preferenza di voto è legalmente considerato un dato sensibile.

Scheda Bianca: Possibilità di procedere a una votazione completa e riconosciuta senza esprimere nessuna preferenza.

Totem: Dispositivo fisico attraverso il quale gli utenti potranno registrarsi e utilizzare il sistema di voto. Ogni seggio elettorale conterrà più totem in base all'affluenza stimata.

1.5 Requisiti funzionali:

I requisiti che specificano gli aspetti funzionali del software. Possono essere visti come un approfondimento dei concetti generali visti nei punti [0.1](#) e [1.3](#). Devono essere espressi in linguaggio semplice e non troppo tecnico.

- Il sistema deve permettere all'utente che lo utilizza di autenticarsi, in modo da distinguere a quale gruppo/categoria di utenti esso appartiene
- Il sistema deve supportare due diversi metodi di autenticazione: tramite nome utente e password, e tramite identità digitale SPID.
- Il sistema deve supportare tipologie di voto multiple, tra le quali voto ordinale, categorico o referendum.
- Deve essere possibile per tecnici specifici poter configurare il software per predisporlo alla votazione
- Il sistema deve essere in gradi di fornire i risultati delle votazioni, esclusivamente dopo il termine delle stesse.
- Il software permetterà soltanto a cittadini aventi diritto di voto autenticati tramite SPID di votare, e prima di convalidare il voto chiederà conferma all'utente per la sua preferenza data.
- Il sistema dovrà essere dotato di opzioni di utilizzo facilitato per utenti ipovedenti o limitati nei movimenti.

1.5.1 Requisiti funzionali utente:

- All'utente votante deve essere consentito identificarsi mediante SPID senza necessità di essersi registrato al sistema di voto.
- L'utente votante potrà richiedere assistenza in caso di difficoltà o disservizio tecnico selezionando l'apposita icona di aiuto.
- L'utente votante potrà sempre avere la possibilità di registrare il proprio voto senza esprimere nessuna preferenza (scheda bianca)
- Tecnici abilitati potranno loggarsi e avere accesso alle impostazioni del sistema, quali verificare che il dispositivo sia funzionante e collegato alla rete.
- Agli utenti votanti dovrà essere garantita un'interfaccia utente sufficientemente semplice ed intuitiva da permettere di ottenere margini di

errori da parte dell'utente finale entro i limiti del 10%, a seguito di un breve video informativo.

- Agli utenti dovrà essere messa a disposizione la possibilità di attivare l'interfaccia con contrasto elevato per chi ha problemi di vista
- Il Responsabile Generale deve aver la possibilità di impostare una nuova votazione, monitorarla e pubblicarne i risultati

1.6 Requisiti non funzionali:

I requisiti non funzionali rappresentano i vincoli e le proprietà/caratteristiche relative al sistema, come vincoli di natura temporale, vincoli sul processo di sviluppo e sugli standard da adottare. Non si applicano quindi a singole funzioni o servizi ma all'intero sistema. Sono poi ulteriormente suddivisibili in base alle macroaree che coprono, come quelli esterni, di processo o prodotto e quelli relativi alla sicurezza.

Alcuni requisiti non funzionali di prodotto individuati:

- Affidabilità: il software, data la natura delicata del suo compito, deve comportarsi come previsto. La probabilità di assenza di fallimenti/malfunzionamenti in 1 ora di utilizzo su una postazione di voto deve essere molto alta (>95%). La causa più probabile di fallimento stimata sarà causata da un utilizzo non corretto da parte dell'utente, e non da malfunzionamenti interni.
- Efficienza: è indispensabile che il software sfrutti nella maniera più efficiente possibile le risorse hardware dei dispositivi utilizzati per il voto, e del database centrale. Essendo il sistema utilizzabile per votazioni non solo locali ma anche statali, il picco di richieste durante queste votazioni è previsto essere alto. Un utilizzo non efficiente dell'hardware in questi casi può portare al blocco del sistema.
- Usabilità: si identificano due possibili expected user:

ELETTORE: utente con conoscenze informatiche ipotizzabili come molto basse, che ha seguito un breve video dimostrativo sulla modalità di voto. Richiesta altissima facilità d'uso, con schermate contenenti meno elementi possibili e molto intuitive.

TECNICO: utente esperto che ha superato corsi di formazione specifici riguardanti il software in uso. Richiesto l'accesso a decine di parametri configurabili, a scapito della facilità d'uso, e la possibilità di riavviare o riconfigurare i dispositivi.

La terza tipologia di attore individuata nelle fasi precedenti, il RESPONSABILE GENERALE, non viene considerata in questo requisito. Questo perché tale figura non interagisce direttamente col sistema software delle postazioni, ma avrà un accesso specifico direttamente al sistema di gestione centrale.

- Robustezza: in circostanze non previste, il software deve assicurarsi che la preferenza (voto) espresso dall'utente non vada perso. Qualora vi siano problemi nella connessione alla rete/database, la votazione da parte di quel dispositivo deve essere immediatamente sospesa.
- Facilità di manutenzione: necessaria per permettere ai tecnici di intervenire in caso di malfunzionamento anche durante le votazioni, permettendo un tempo di intervento ridotto, stimato entro i 30 minuti.
- Portabilità: prevedendo la possibilità che i dispositivi hardware vengano successivamente aggiornati o sostituiti, il software deve poter essere installato e supportare sistemi diversi.

1.6.1 Requisiti esterni:

Requisiti normativi: un utente votante deve poter esprimere il proprio voto una sola volta per ogni diversa votazione. Per nessuna ragione il voto deve poter essere espresso due volte, oppure modificato in seguito. In caso di tentativo di secondo voto, il sistema dopo l'autenticazione provvederà a impedire qualsiasi operazione da parte dell'utente.

Il sistema di voto deve rispettare totalmente le normative di voto imposte dalla legge.

Requisiti etici e legali: Il sistema non deve inserire opzioni di voto differenti da un sistema di voto cartaceo, per evitare ad esempio differenze col voto espresso tramite posta.

Non deve essere possibile accedere ai risultati prima del termine delle votazioni (exit poll) per evitare di influenzare la rimanente platea votante.

Ogni malfunzionamento rilevato durante le votazioni deve essere sempre segnalato, in modo che possa essere analizzato a posteriori e in base alla valutazione determinare se la votazione è da considerarsi valida (sistema di auditing).

1.6.2 Requisiti di sicurezza:

- **Prevenzione:** il sistema per sua natura è fortemente soggetto ad attacchi, volti a bloccare il suo funzionamento, ad esempio sovraccaricando di richieste il sistema, o volti a modificare le elezioni, alterando i voti registrati.
- Il sistema deve essere in grado di rilevare operazioni non autorizzate anche effettuando auditing in tempo reale sui dispositivi di voto e sull'analisi dei dati inviati dai dispositivi al server centrale. Deve essere possibile rilevare un utilizzo non riconducibile a un utente reale ma a un programma automatizzato di voto e segnalare tale situazione al Responsabile Generale.
- **Tracciabilità:** ogni operazione, tra le quali l'autenticazione tramite SPID o l'accesso di un tecnico alle impostazioni di sistema, deve essere registrata in appositi log di sistema. Per ogni operazione deve essere indicato l'orario, l'utente loggato che la esegue, l'esito dell'operazione e il dispositivo fisico dal quale viene inviato il comando. Tali dati di log dovranno essere salvati e dotati di un sistema di backup esterno e separato dal backup generico, in modo da poter salvare i dati a seguito di un attacco riuscito al sistema.
- **Privacy:** l'utente deve essere informato riguardo al modo in cui verranno gestiti i suoi dati. Deve essere garantito che il sistema si assicuri attraverso monitoraggio degli accessi, a quali utenti viene permesso di accedere ai dati degli utenti
- **Sicurezza a più livelli:** i dati gestiti dal sistema non hanno tutti lo stesso grado di riservatezza. Mentre alcuni dati, come quelli anagrafici, devono poter essere letti da tecnici certificati per effettuare verifiche, altri dati come la preferenza espressa al voto non devono in alcun modo essere accessibili e riconducibili a uno specifico utente.
- **Autenticazione:** l'accesso di un utente votante avverrà mediante identificazione digitale SPID. L'accesso di un tecnico specializzato, che è un'operazione potenzialmente pericolosa per il sistema, deve avvenire mediante combinazione di credenziali di accesso e codice di conferma univoco e temporaneo tramite sms.

Ogni utente votante deve poter effettuare il logout in qualsiasi momento durante il suo utilizzo del software.

- Robustezza: il software di voto dovrà essere progettato tenendo conto della possibilità di brogli elettorali da parte dei singoli elettori che potrebbero tentare di utilizzarlo in maniera diversa da quanto progettato. Il software dovrà quindi essere progettato per tenere specifici comportamenti in circostanze di utilizzo non previste.

1.7 User stories:

I seguenti esempi sono basati sui requisiti funzionali e utente dei punti [1.5](#) e [1.5.1](#). Rappresentano quindi solo un'applicazione diretta dei requisiti e non coprono tutte le possibili applicazioni.

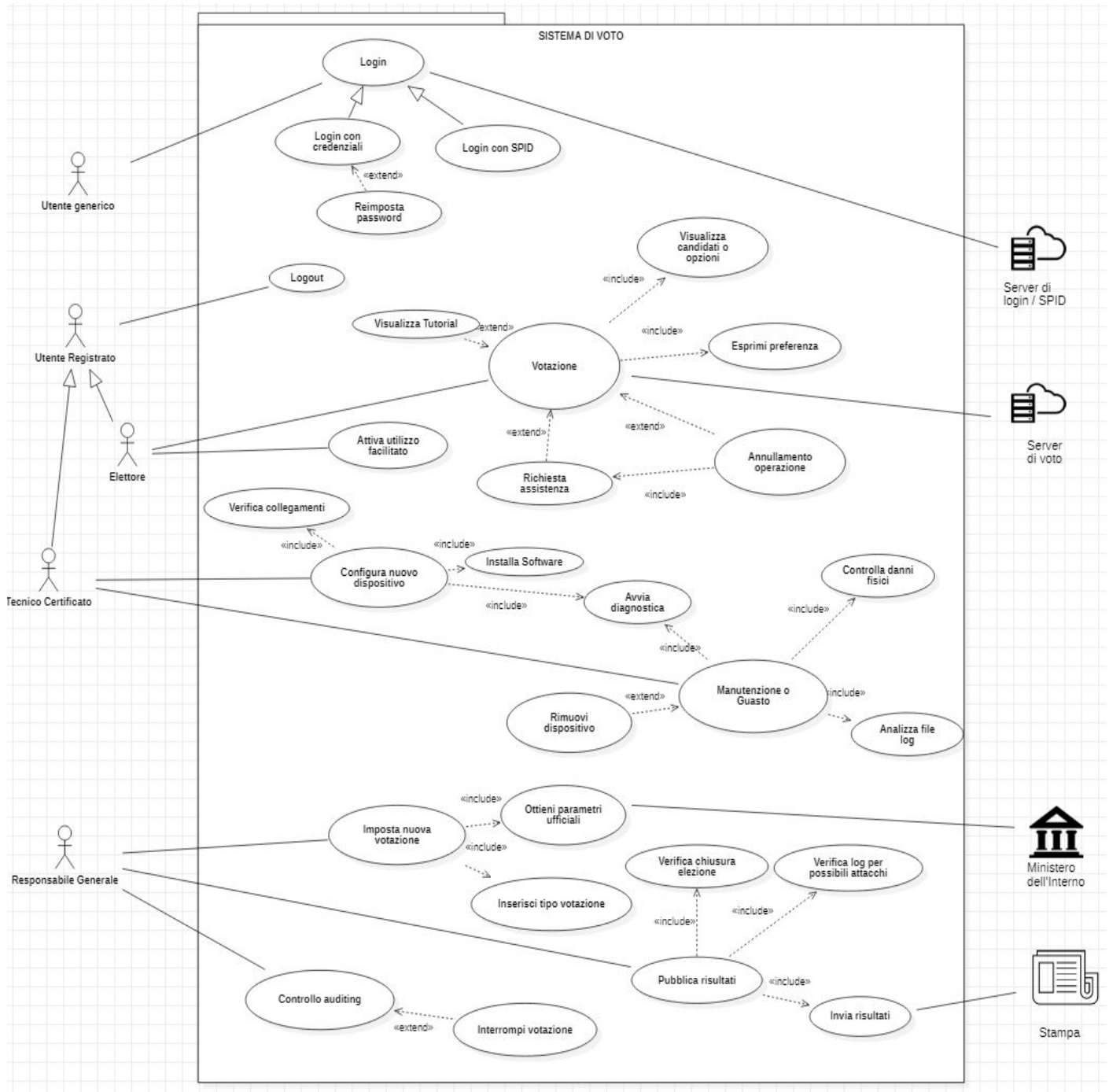
Per una maggior precisione e maggiori dettagli vedere il capitolo Use Case con i relativi scenari d'uso.

Alcuni esempi:

1. Come cittadino voglio potermi identificare per accedere al voto attraverso la mia identità digitale, in maniera autonoma senza dover richiedere assistenza.
2. Come cittadino, dopo essermi identificato, voglio che il sistema automaticamente mi permetta di accedere a tutte le votazioni alle quali ho il diritto di voto, basandosi sulla mia residenza. (Esempio elezioni statali generali e in contemporanea elezioni comunali per un ristretto numero di comuni) senza che mi vengano mostrate votazioni per le quali non sono autorizzato.
3. Come tecnico certificato voglio poter accedere a funzioni riservate ad amministratori del sistema per poter configurare sul posto i dispositivi, o per monitorare l'andamento ed eventuali malfunzionamenti in corso.
4. Come tecnico, voglio poter disporre di un login per accedere a funzioni da amministratore, e di un login per poter esercitare il mio diritto di voto da cittadino.
5. Come cittadino, voglio essere certo che nessun risultato parziale (exit poll) venga pubblicato prima del termine delle votazioni.
6. Come cittadino, voglio avvalermi del diritto all'anonimato in modo che non sia possibile ricondurre a me la preferenza scelta.

2 Progettazione del sistema

2.1 Use Case



Si ricorda che la cartella immagini contiene gli screenshot originali dei diagrammi

Descrizione dei principali elementi del diagramma:

Gli attori Utente generico e Utente registrato sono stati ideati per rappresentare meglio la condizione di una persona prima e dopo che venga identificata dal sistema di voto. Suddivisione molto utile anche per il successivo diagramma delle classi.

I tre principali attori, già identificati nelle [Specifiche aggiuntive](#) e negli Expected User ([1.6](#)). L'Elettore può iniziare la procedura di voto, all'interno della quale sono presenti diversi passaggi anche facoltativi. Oppure può richiedere la modalità facilitata. Il Tecnico certificato può configurare un nuovo dispositivo (normalmente prima delle votazioni) o riparare uno guasto (per il quale ha un limite di tempo). Entrambi questi attori derivano dal super-attore Utente registrato. Questo perché vista la sicurezza che deve avere il sistema, a nessun utente dubbio (non registrato) deve essere permesso fare nulla se non autenticarsi. Infine l'attore Responsabile Generale viene considerato a parte per due motivi: non necessita di loggarsi ai totem perché autorizzato ad accedere direttamente al server fisico, e non si esclude la possibilità che quell'attore possa in futuro venire rappresentato non da una singola persona ma da un apposito ufficio.

Nel lato destro sono riportate le entità che non rappresentano attori o che non intervengono in maniera attiva. Queste entità verranno riprese in maniera più approfondita nei successivi diagrammi di sequenza e attività, motivo per cui è stato ritenuto opportuno inserirli già dall'inizio.

Elenco di alcune condizioni necessarie per il verificarsi delle <<extend>>:

<<extend>>Reimposta password: click da schermata di login a condizione che l'username sia corretto. Richiede una serie di passaggi per verificare l'identità e la successiva scelta di una nuova password.

<<extend>>Visualizza tutorial: solo se l'utente seleziona l'icona assistenza, e poi tutorial.

<<extend>>Annulla operazione: eseguibile in qualsiasi momento della procedura, selezionando l'icona di chiusura. Si suppone che il mancato completamento del voto sia a causa di un problema di comprensione dell'utente; quindi si include la richiesta di assistenza.

Più avanti nel diagramma...

<<extend>> Interrompi votazione: si attiva dopo la verifica auditing e dei log, e viene richiesta solo ed esclusivamente se viene rilevato un broglio, un errore dei server o un attacco informatico.

2.2 Scenari

Ai fini della tracciabilità all'interno della relazione, ad ogni scenario viene associato un indice di capitolo, in modo che anche nei diagrammi successivi sia possibile reperire con facilità lo scenario associato.

Gli scenari inoltre sono riconducibili ad alcuni requisiti richiesti (diversi a seconda dello scenario) e fanno tutti riferimento al diagramma Use Case.

2.2.1 Scenario 1 (Utente generico login)

L'utente si avvicina al totem e avvia la sequenza di login, scegliendo se identificarsi tramite SPID (se è un Elettore) o tramite credenziali (se è un tecnico certificato).

L'utente inserisce credenziali corrette e perciò viene autenticato. Se l'utente è un tecnico può voler reimpostare la password.

Tracciabilità requisiti funzionali (cap. [1.5](#)):

- Il sistema deve permettere all'utente che lo utilizza di autenticarsi, in modo da distinguere a quale gruppo/categoria di utenti esso appartiene
- Il sistema deve supportare due diversi metodi di autenticazione: tramite codice fiscale e password, e tramite identità digitale SPID.

Nome	Autenticazione
Scopo	Permettere agli utenti di identificarsi
Attori	Cittadini votanti e personale tecnico
Precondizioni	Dispositivi accesi e connessi alla rete
Trigger	Utente clicca su login
Sequenza eventi	Scelta tipologia utente -> inserimento credenziali -> verifica credenziali -> utilizzo sistema
Postcondizione	Se l'utente ha credenziali valide deve poter accedere
Alternativa	Credenziali errate comporta il ripetere il login

2.2.2 Scenario 2 (Utente ipovedente)

Un qualsiasi Elettore che soffre di problemi di vista è in grado di attivare in qualsiasi momento la modalità facilitata per avere schermate con contrasto elevato e caratteri più grandi.

Tracciabilità con requisiti [1.5](#):

- Agli utenti dovrà essere messa a disposizione la possibilità di attivare l'interfaccia con contrasto elevato per chi ha problemi di vista
- Agli utenti votanti dovrà essere garantita un'interfaccia utente sufficientemente semplice ed intuitiva da permettere di ottenere margini di errori da parte dell'utente finale entro i limiti del 10%, a seguito di un breve video informativo.

(Si ipotizza che la mancata attivazione del contrasto elevato per cittadini ipovedenti possa portare a maggiori errori di utilizzo)

Nome	Modalità facilitata
Scopo	Consentire l'utilizzo del sistema a utenti ipovedenti
Attori	Cittadini votanti
Precondizioni	Dispositivi accesi e connessi alla rete, login effettuato
Trigger	Utente attiva la modalità facilitata
Sequenza eventi	Click su modalità ipovedenti -> modifica della schermata di voto -> proseguimento della procedura di voto
Postcondizione	Attivazione contrasto elevato e caratteri più grandi
Alternativa	Non previste

2.2.3 Scenario 3 (Guasto del dispositivo di voto)

Un tecnico certificato, avvisato del guasto di un dispositivo, procede con la riparazione. Verifica se sono presenti danni fisici come rotture da caduta, surriscaldamento o cavi staccati, controlla i file di log ed effettua una diagnosi. Se il guasto viene individuato lo ripara, altrimenti rimuove il dispositivo.

Tracciabilità con requisiti [1.5](#) , non solo utente:

- Tecnici abilitati potranno loggarsi e avere accesso alle impostazioni del sistema, quali verificare che il dispositivo sia funzionante e collegato alla rete
- Facilità di manutenzione: necessaria per permettere ai tecnici di intervenire in caso di malfunzionamento anche durante le votazioni, permettendo un tempo di intervento ridotto, stimato entro i 30 minuti.
- Tracciabilità: ogni operazione, tra le quali l'autenticazione tramite SPID o l'accesso di un tecnico alle impostazioni di sistema, deve essere registrata in appositi log di sistema. Per ogni operazione deve essere indicato l'orario, l'utente loggato che la esegue, l'esito dell'operazione e il dispositivo fisico dal quale viene inviato il comando. Tali dati di log dovranno essere salvati e dotati di un sistema di backup esterno e separato dal backup generico, in modo da poter salvare i dati a seguito di un attacco riuscito al sistema.

Nome	Guasto dispositivo
Scopo	Ripristinare il dispositivo attraverso intervento tecnico
Attori	Tecnico certificato
Precondizioni	Tecnico certificato disponibile all'intervento
Trigger	Segnalazione guasto tecnico
Sequenza eventi	Ricezione segnalazione guasto -> tentativo di login del tecnico -> check guasti fisici -> check log -> riparazione
Postcondizione	Dopo l'intervento, assenza di dispositivi guasti nella sala di voto
Alternativa	Se la riparazione non è effettuabile in loco, rimozione dispositivo

2.2.4 Scenario 4 (Pubblicazione risultati)

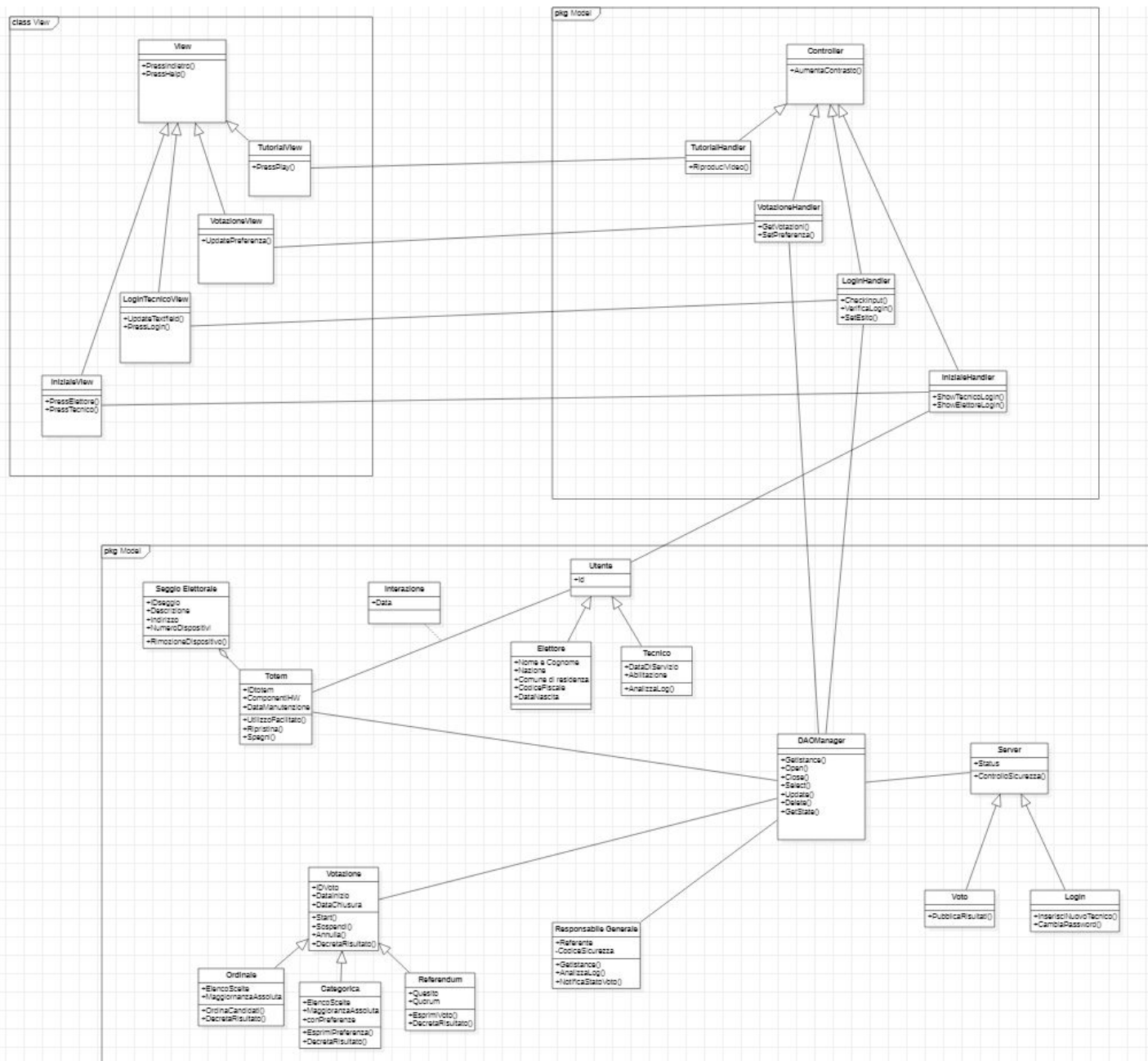
Il Responsabile Generale interviene direttamente sul sistema lato server per richiedere la pubblicazione dei risultati. Prima di confermare il comando, verifica i log generali per assicurarsi non si siano verificati guasti, problemi o attacchi.

Tracciabilità requisiti [1.5](#):

- Il sistema deve essere in grado di rilevare operazioni non autorizzate anche effettuando auditing in tempo reale sui dispositivi di voto e sull'analisi dei dati inviati dai dispositivi al server centrale. Deve essere possibile rilevare un utilizzo non riconducibile a un utente reale ma a un programma automatizzato di voto e segnalare tale situazione al Responsabile Generale.
- Il Responsabile Generale deve aver la possibilità di impostare una nuova votazione, monitorarla e pubblicarne i risultati
- Non deve essere possibile accedere ai risultati prima del termine delle votazioni (exit poll) per evitare di influenzare la rimanente platea votante.

Nome	Pubblicazione risultati
Scopo	Rendere pubblici i risultati delle elezioni
Attori	Responsabile generale, Ministero, Servizi di informazione
Precondizioni	Deve esistere la votazione per la quale viene richiesto il risultato
Trigger	Richiesta risultati
Sequenza eventi	Responsabile richiede risultati -> controllo file log per brogli -> controllo termine elezione -> pubblicazione risultati a Ministero e Servizi di Informazione
Postcondizione	Pubblicazione dei risultati
Alternativa	Se la votazione non è terminata, la richiesta viene annullata

2.3 Diagramma delle classi



Descrizione:

La creazione del diagramma delle classi ha richiesto solo nella fase più iniziale il supporto del diagramma [Use Case](#), utilizzato come punto di partenza.

Quello che più di tutti ha portato al completamento del diagramma delle classi è stato lo studio dei diagrammi di [sequenza](#) basati sugli [scenari](#), che hanno mostrato quali sono le interazioni tra gli oggetti.

Si noti come la figura del Responsabile Generale corrisponda a quanto indicato nelle [considerazioni iniziali](#): una figura esterna all'utente generico, che accede direttamente al server e che presenta attributi diversi da un utente generico, a dimostrazione che tale figura non vada intesa come "persona" ma come figura quasi istituzionale. Si ipotizza che l'autenticazione del responsabile direttamente sul server può avvenire con credenziali speciali o, più verosimilmente, attraverso dati biometrici e con la garanzia che l'accesso al server sia gestito sotto l'osservazione dei servizi di sicurezza interni all'edificio. Pertanto l'accesso fisico alla sala server sarà sorvegliato da istituti di vigilanza che ne consentiranno l'accesso solo a personale conosciuto e certificato.

Il diagramma è stato infine rifinito applicando i principi base dei Design Pattern. Per maggiori dettagli vedere il capitolo [3.1](#).

Valutazione delle proprietà del diagramma delle classi:

High Cohesion: proprietà che indica quanto sono logicamente correlati i dati e le operazioni all'interno di una classe. Serve a indicare se una classe è altamente specializzata o se invece è stata ideata come troppo generica, contenendo metodi con funzioni o domini di applicazioni troppo diversi e variegati tra loro.

Nel caso di questo sistema, le classi risultano avere una coesione molto alta, essendo ognuna di esse specializzata per compiti precisi. Alcuni esempi a supporto possono essere la classe Responsabile Generale, creata appositamente per rappresentare una figura specializzata, e per la quale è stato scelto di non farla derivare da una generalizzazione della classe Utente Generico, dalla quale avrebbe ereditato attributi e metodi non strettamente correlati alla sua funzione.

Low Coupling: proprietà che indica quanto le classi sono dipendenti tra di loro, misurando e valutando le connessioni e le relazioni tra di loro, visibili anche nel diagramma delle classi. Normalmente, avere un accoppiamento alto significa non aver assegnato correttamente le responsabilità alle classi.

In questo sistema, l'accoppiamento è sufficientemente basso, anche se migliorabile ulteriormente. Alcuni elementi che tendono ad "alzarlo" sono:

La relazione Responsabile Generale -> Server di voto -> Votazione. L'assegnare una possibile relazione diretta tra Responsabile Generale e Votazione diminuirebbe ulteriormente l'accoppiamento, ma a scapito di una maggior sicurezza che viene fornita accedendo al server per poter operare sulla votazione.

Le relazioni tra la macroarea View e Model passano attraverso le classi Controller. Questo, nonostante aumenti l'accoppiamento, è stato ritenuto indispensabile visti i vantaggi portati dall'applicazione del Design Pattern MVC, trattati nel capitolo [3.1](#).

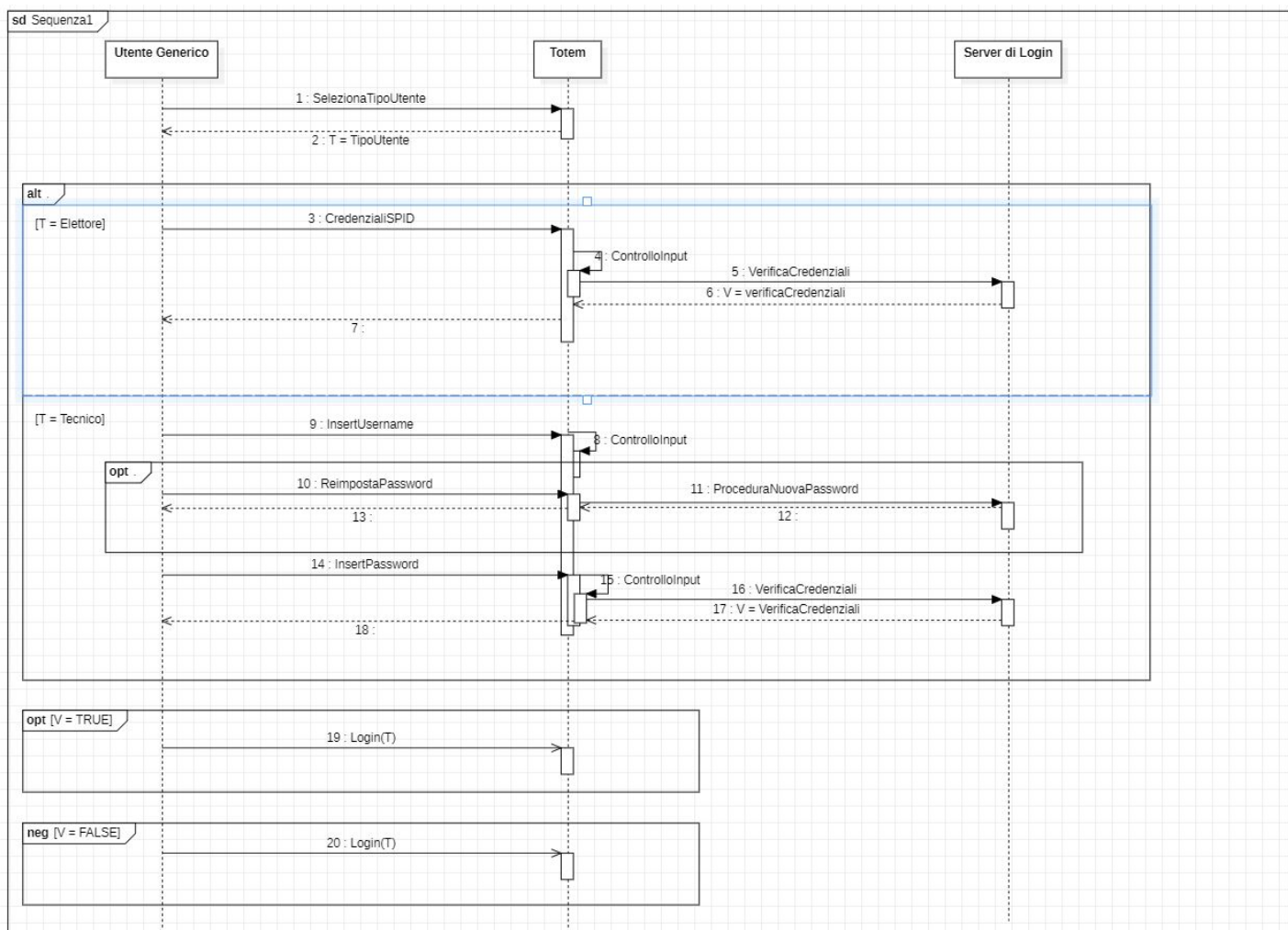
2.4 Diagrammi di sequenza

I seguenti diagrammi di sequenza sono riferiti ai 4 possibili scenari descritti nel capitolo [2.2](#). Ogni diagramma fa riferimento a uno scenario specifico. La tracciabilità generale è da intendersi in questo ordine:

Requisiti -> Scenari riferiti ai precedenti requisiti e Use Case -> Diagrammi di sequenza riferiti ai precedenti scenari.

Sequenza scenario 1:

Riferimento allo [Scenario 1](#) (Utente generico login)



Descrizione:

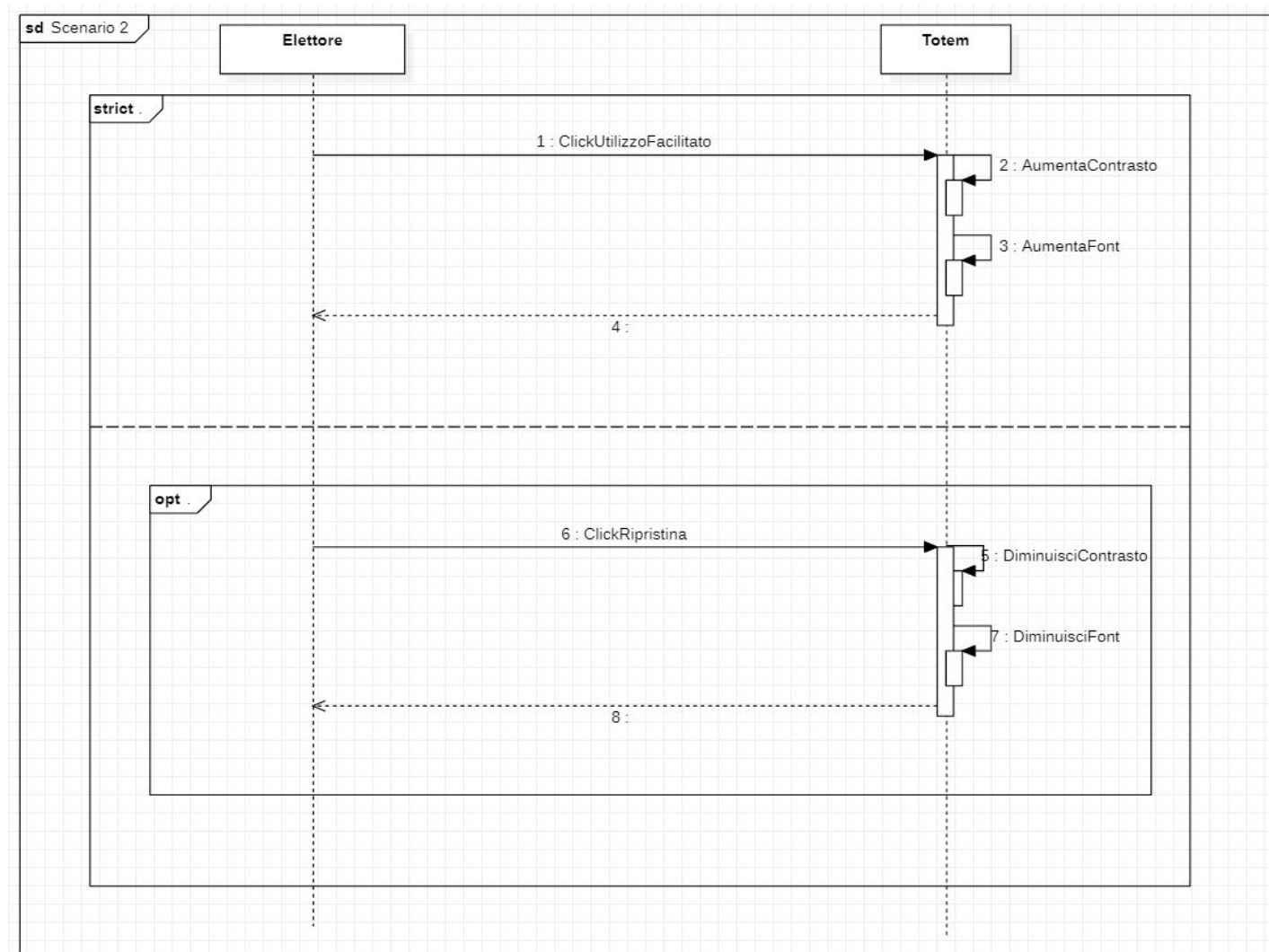
L'utente seleziona attraverso il totem quale tipo di login vuole eseguire, se come Elettore o se come tecnico (vedere le schermate della GUI per maggiori dettagli). Il cittadino inserisce le credenziali SPID, e il totem verifica sui server se esiste tale utente.

Il tecnico si autentica tramite credenziali, e se lo desidera può avviare un reimposta password.

Al termine, se il server di autenticazione fornisce esito positivo l'utente chiede al totem di essere riconosciuto come tipo di utente T (Votante o Tecnico). E' fondamentale che se l'esito del login risulti essere negativo, l'utente non debba essere riconosciuto.

Sequenza scenario 2:

Riferimento allo [Scenario 2](#) (Utente ipovedente)



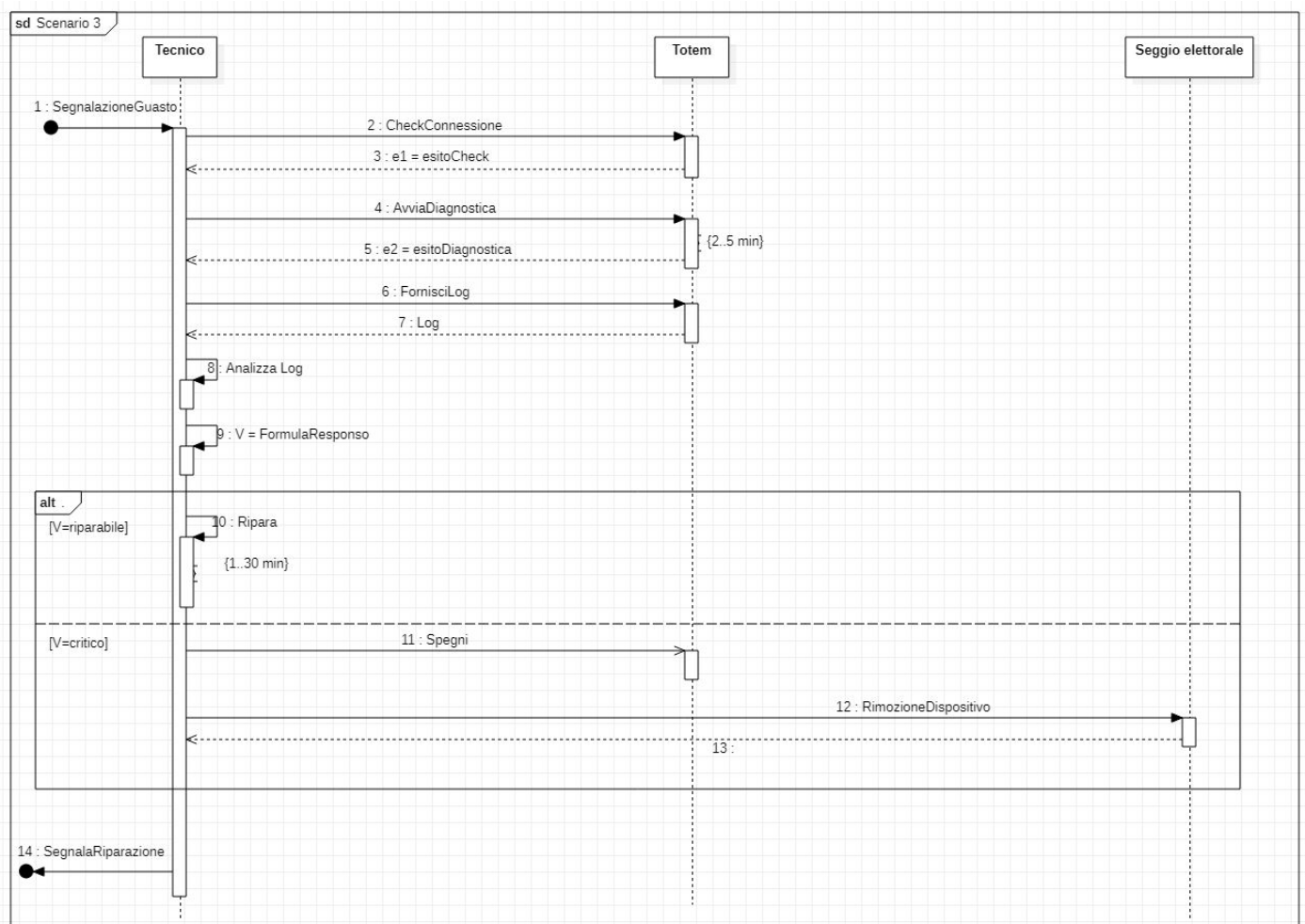
Descrizione:

Un Elettore che soffre di problemi alla vista, e che perciò ha difficoltà a proseguire con la votazione, clicca sull'apposito pulsante per attivare la modalità facilitata. Il totem incrementa il contrasto dello schermo e le dimensioni dei caratteri.

Se la modalità è stata attivata per sbaglio, e solo dopo che la modalità viene attivata, è possibile annullare tutte le modifiche cliccando sul pulsante ripristina.

Sequenza scenario 3:

Riferimento allo [scenario 3](#) (Guasto al totem)



Descrizione:

Un Tecnico certificato riceve (dal sistema, dal seggio, o a voce dall'utente) la segnalazione di un guasto ad un totem. Il tecnico interagisce col totem eseguendoci

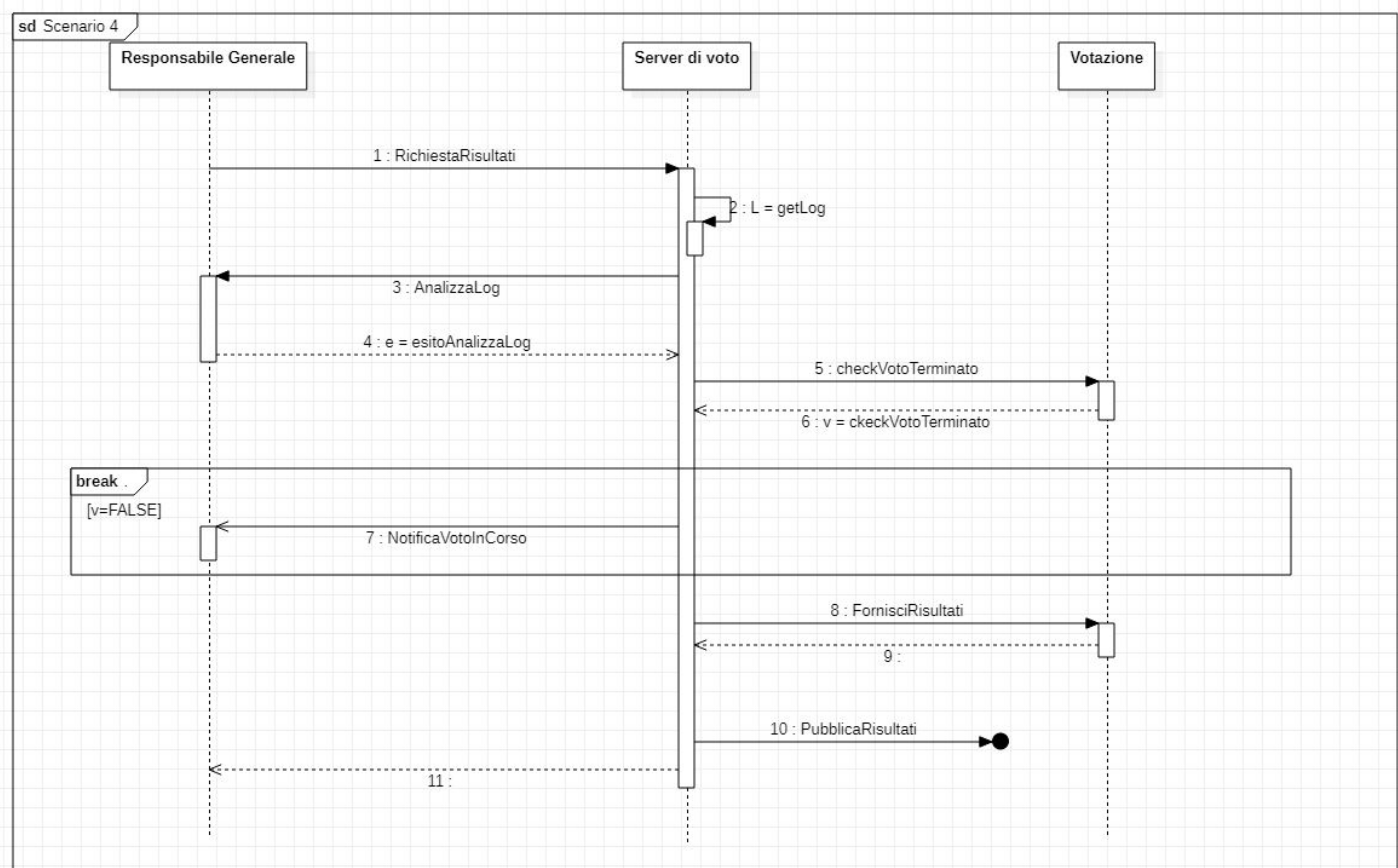
un test di connessione, una diagnostica e scaricandone i Log. Il tecnico analizza i risultati e valuta quale può essere il problema.

Se è riparabile entro 30 minuti (vedere requisito di Facilità di Manutenzione, cap 1.6) esegue la riparazione in loco. Altrimenti disattiva il totem e segnala al sistema del seggio elettorale che da ora vi è un totem funzionante in meno.

Ai fini di tracciabilità e completezza dei Log, il tecnico comunica la chiusura della riparazione.

Sequenza scenario 4:

Riferimento allo [scenario 4](#) (Pubblicazione risultati)



Descrizione:

Il Responsabile generale richiede la pubblicazione dei risultati al server di voto. Il server fornisce al responsabile i file di log che vengono controllati per vedere se evidenziano problemi o attacchi, poi verifica se la votazione in questione sia terminata. Se non lo è lo notifica al responsabile e termina. Se è finita, recupera i dettagli e i risultati della votazione e li invia agli organi competenti, visibili nel diagramma [UseCase](#).

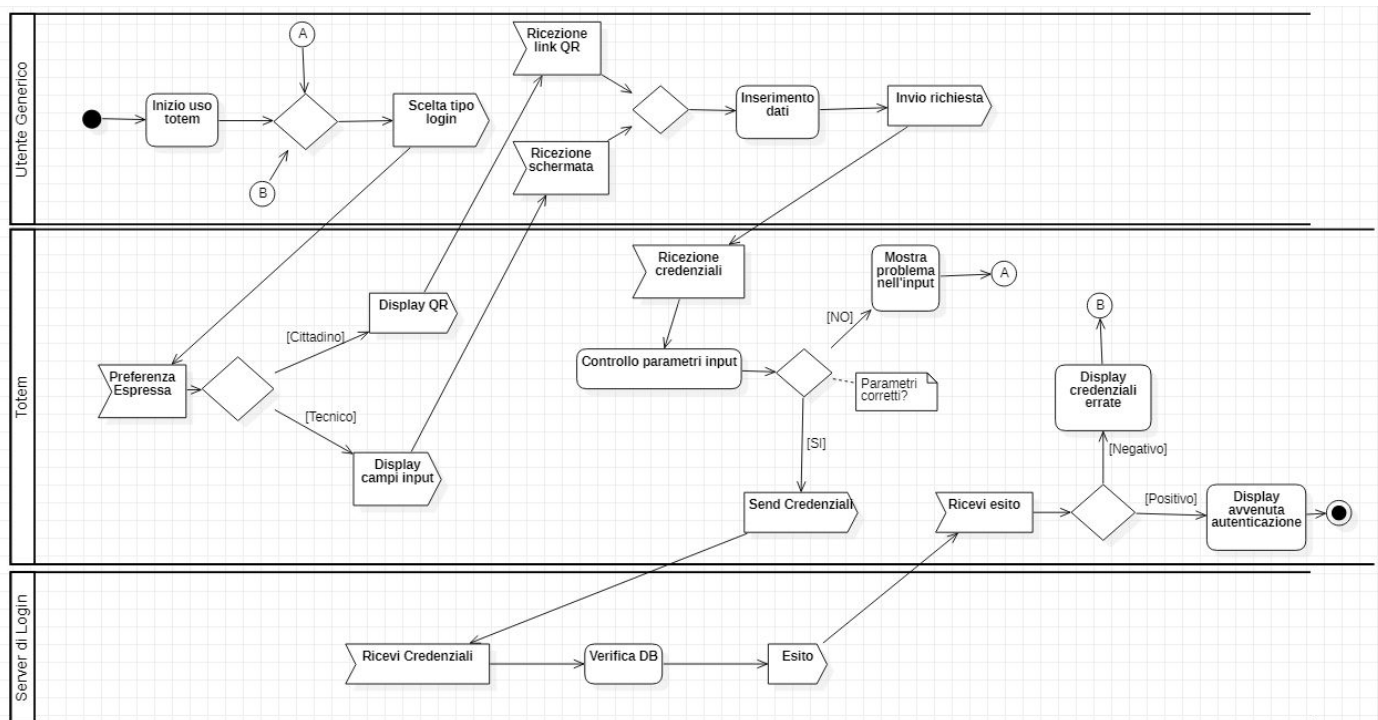
2.5 Diagrammi di attività

Qui di seguito vengono riportati alcuni diagrammi di Attività. Nelle indicazioni di progetto non è stato richiesto di realizzare un diagramma di attività per ogni scenario come per i diagrammi di sequenza, tuttavia gli esempi presi in considerazione per i seguenti diagrammi prendono comunque spunto dagli scenari del capitolo [2.2](#).

Questo porta ad avere delle descrizioni dei diagrammi di attività ridondanti dato che la situazione è già stata rappresentata e discussa nei capitoli precedenti.

Activity 1

Riferimento allo [Scenario 1](#) di Login.



Le classi coinvolte sono Utente Generico, Totem, Server di Login. Ognuna di esse ha una propria swimline per facilità di lettura e comprensione.

Punti di particolare interesse:

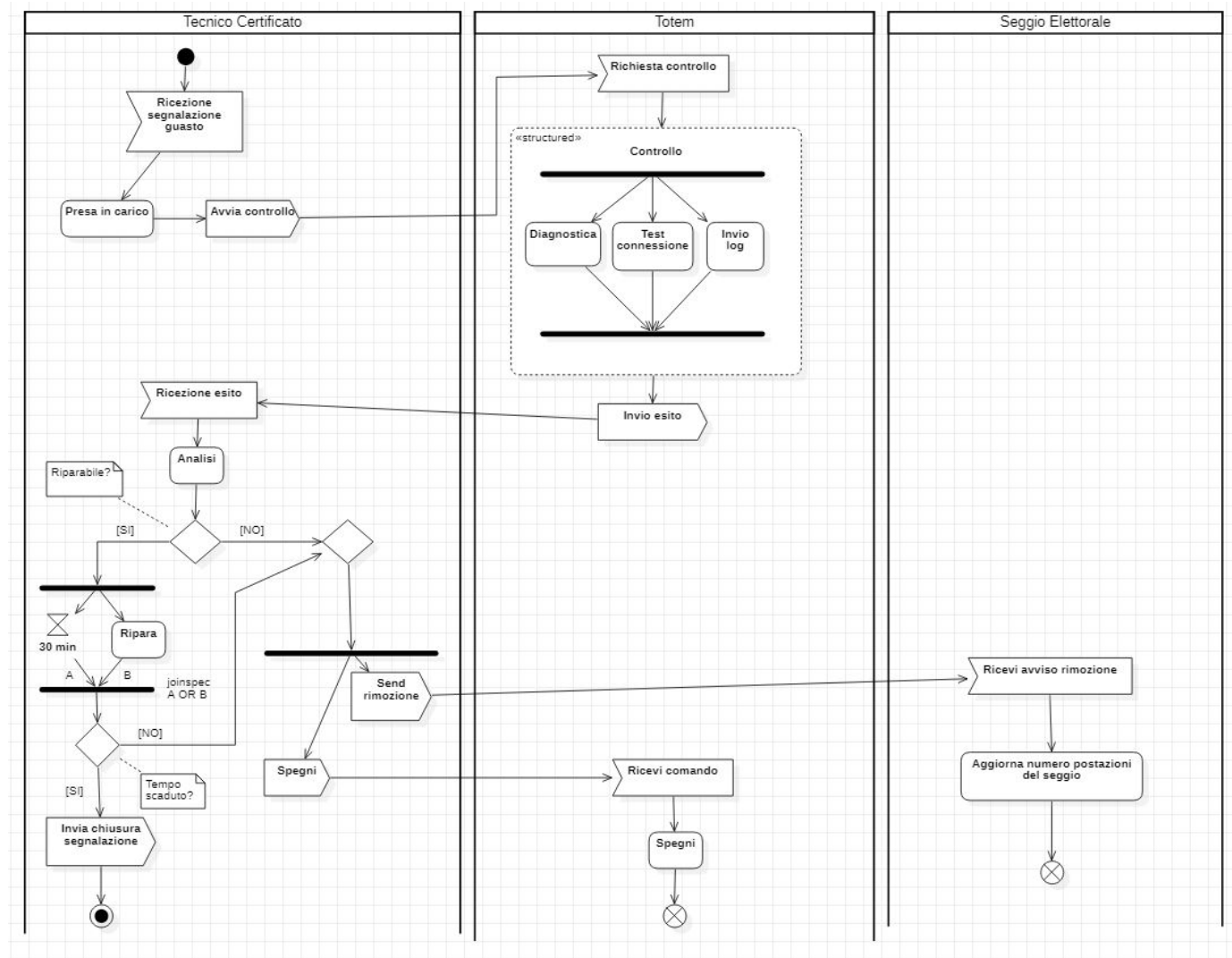
Il primo merge node dopo l'activity "Inizio uso totem" ha lo scopo non solo di raccogliere il flusso del primo utilizzo del totem, ma anche di consentire all'utente di ripetere la procedura nel caso abbia inserito credenziali sintatticamente sbagliate (connector node A) o non presenti nel server (connector node B).

La separazione dei flussi di attività in base alla scelta di login come cittadino o tecnico è una separazione temporanea. Il processo di verifica infatti è unico per entrambi i casi.

Si noti come siano presenti molti send e receive action tra il totem e l'utente. Tutti questi segnali rappresentano soltanto l'aggiornamento della schermata utente visualizzata sul totem. La comunicazione tra totem e server invece è una sola e dovrà essere gestita tenendo presente i requisiti di sicurezza. Verrà inoltre implementata sfruttando il design pattern Adapter.

Activity 2

Riferimento a [Scenario 3](#) guasto del dispositivo di voto.



Le classi coinvolte sono Tecnico certificato, Totem e Seggio elettorale anche se marginalmente.

L'inizio dell'activity gestito dalla ricezione del segnale "Ricevi segnalazione guasto" è compatibile e tracciabile col Found Message corrispondente al relativo Diagramma di sequenza scenario 3.

Il tecnico avvia il controllo, che viene eseguito in ogni sua fase parallela dal totem stesso. In base al requisito Facilità di manutenzione (cap [1.6](#)) Se la riparazione dura più di 30 minuti scade il timer e si passa alla segnalazione di guasto non riparabile.

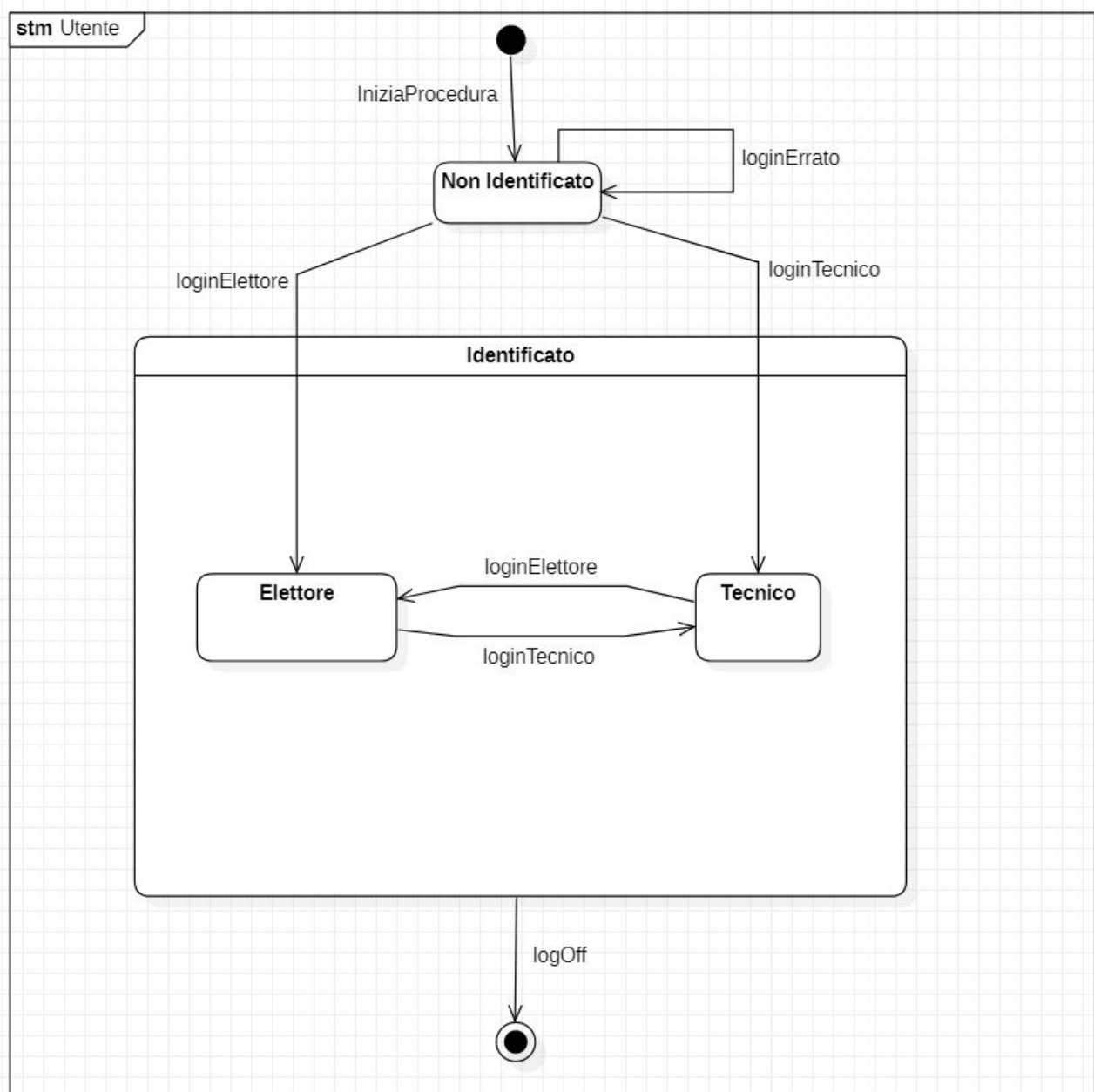
2.6 Diagrammi di stato

Note generali:

Le transizioni che non riportano né eventi né condizioni sulle rispettive frecce sono da intendersi come azioni di completamento, avviate in automatico dall'oggetto al completamento delle operazioni interne a un suo possibile stato.

Gli eventi con i quali sono state indicate alcune transizioni dovranno corrispondere a reali chiamate di metodi nella classe corrispondente all'oggetto preso in considerazione.

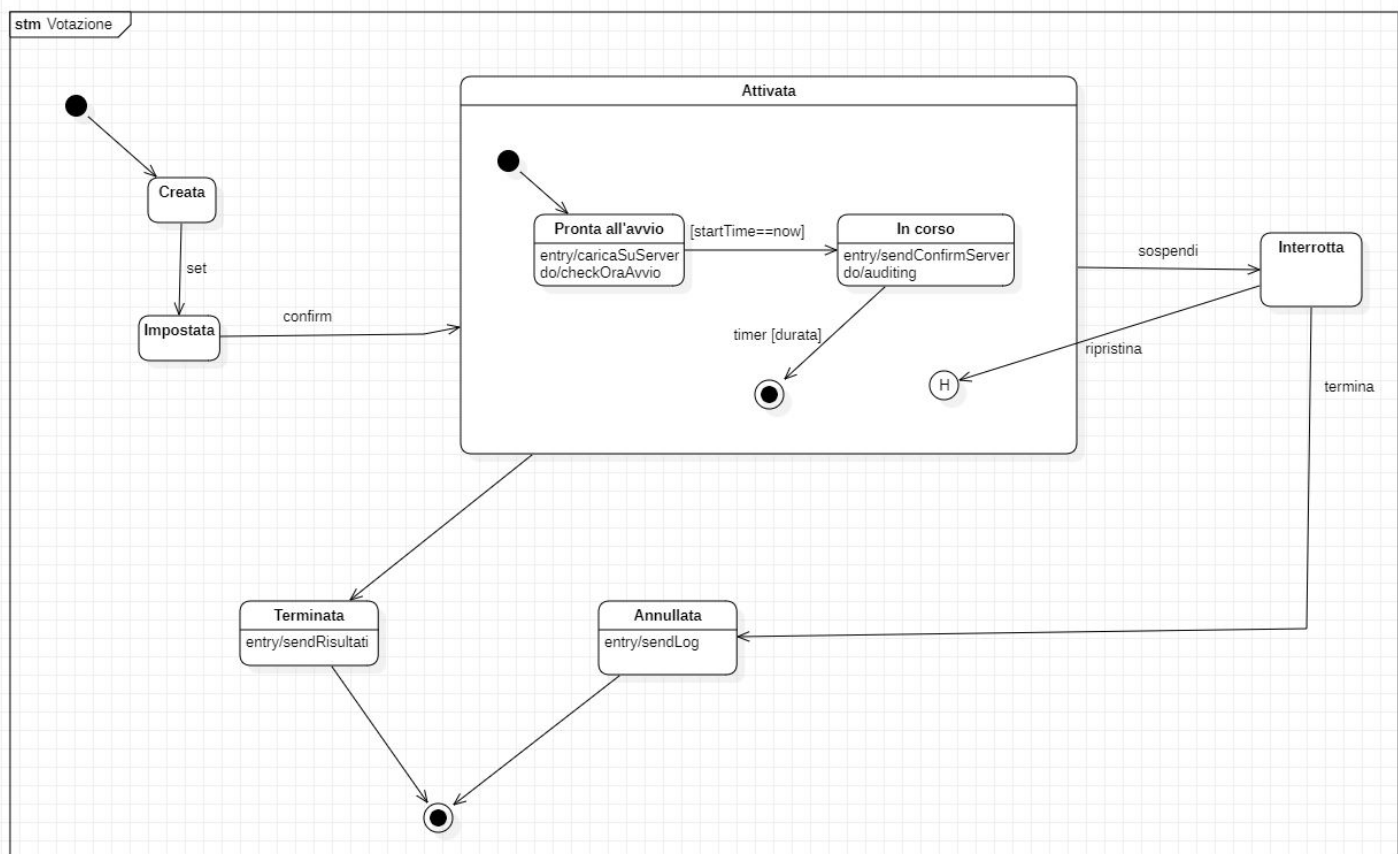
Diagramma di stato UTENTE:



L'oggetto generico Utente viene generato quando inizia un'interazione con il totem. L'utente può rientrare nei due macro-stati "Non identificato" o "Identificato" in base all'aver compiuto o meno la procedura di login. All'interno dello stato Identificato, si ipotizzano due sottostati diversi in base alla tipologia dell'utente, e si ipotizza che sia possibile cambiare sottostato ripetendo il login, pratica probabilmente utile per i test.

L'evento di disconnessione "logOff " porta all'eliminazione dell'oggetto Utente.

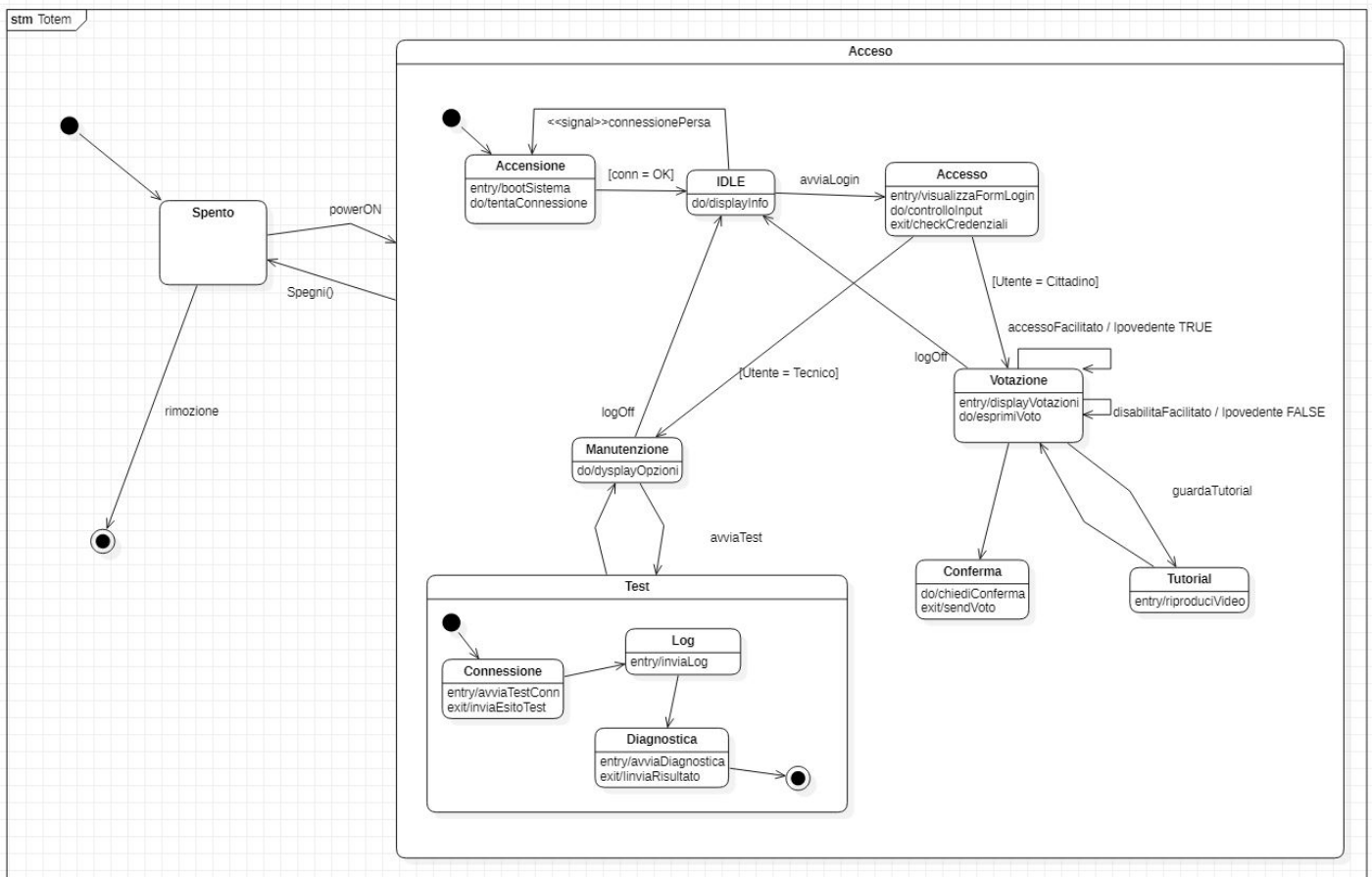
Diagramma di stato VOTAZIONE:



Si noti come l'interruzione di una votazione attivata, che come è stato precedentemente indicato è responsabilità del Responsabile Generale, comporta il passaggio nello stato "interrotta". Mentre si trova in quello stato, l'oggetto singleton Responsabile Generale valuterà in base all'auditing e ai log se il voto può riprendere. In base al risultato della valutazione la votazione riceve il segnale "ripristina", che accede allo storico degli stati per riprendere dal punto in cui era stata interrotta, oppure viene annullata definitivamente.

Si noti come possibili problemi di sicurezza che richiedono una sospensione e successivamente una valutazione possano avvenire in ogni sottostato dello stato generale "attivata", quindi anche quando il voto è configurato ma non ancora in corso.

Diagramma di stato TOTEM:

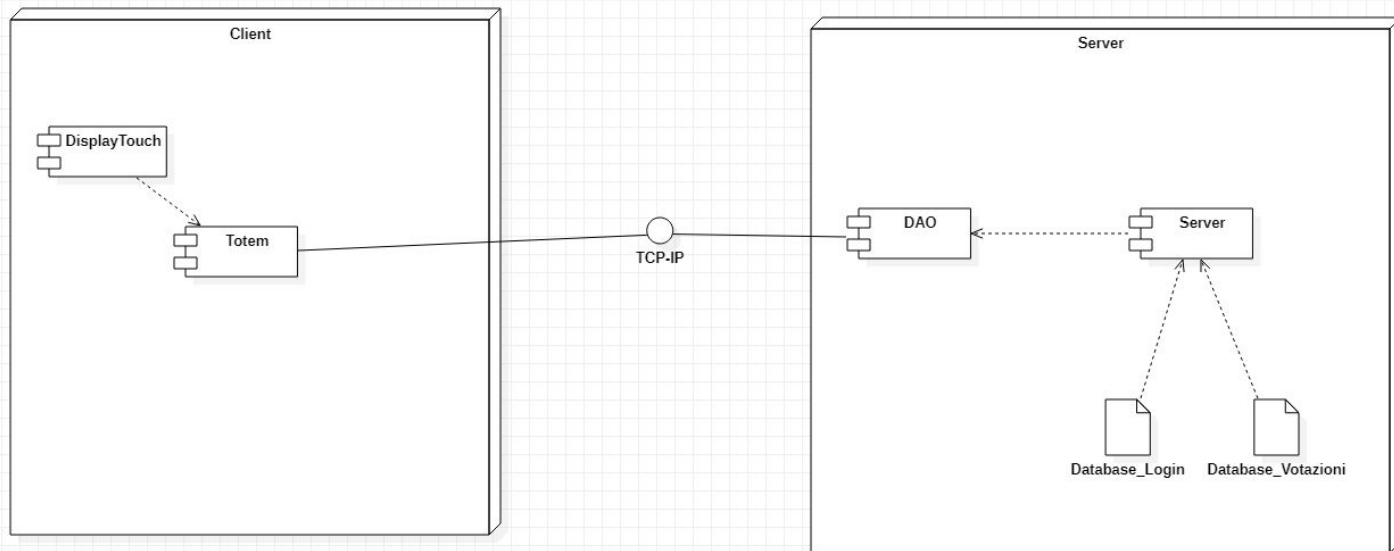


Il totem può trovarsi nello stato Spento o Acceso, e può essere rimosso solo se è Spento.

All'interno dello stato Acceso la procedura iniziale è uguale per entrambi gli utenti. Solo dopo l'avvenuto login il totem può entrare in due stati differenti in base al tipo di utente che lo utilizza.

Qualsiasi sia il sottostato attivo mentre il totem è Acceso, allo scattare dell'evento Spegni il totem esce da ogni sottostato e passa in Spento.

2.7 Diagramma di Deployment



Il diagramma di Deployment descrive il sistema sotto l'aspetto fisico, mostrandone i dispositivi hardware e le relazioni tra di loro.

Nel caso di questo sistema, la componentistica fisica è piuttosto ridotta: escludendo la parte relativa ai seggi elettorali veri e propri, non direttamente legati al progetto, le componenti principali rimanenti sono i dispositivi fisici di voto e il sistema centrale, al quale i Totem si connettono mediante interfaccia internet.

Dal diagramma è possibile notare come il sistema "fisico" rispecchi in pieno il classico paradigma client-server, portando così a giustificare ulteriormente la necessità di implementare il Design Pattern DAO.

3 Implementazione del sistema

3.1 Design Pattern

Nota per la lettura: l'introduzione dei design pattern ha rivoluzionato fortemente la rappresentazione schematica del sistema, in particolare alcuni diagrammi UML. Per motivi di logica di raggruppamento il diagramma delle classi è stato inserito insieme agli altri diagrammi UML nel capitolo 2, ciò nonostante per la lettura completa e corretta di questo capitolo è necessario prendere ancora come riferimento il diagramma delle classi del capitolo [2.3](#) in cui si vedono gli effetti dell'applicazione dei design pattern.

3.1.1 MVC

Il design Pattern Model-View-Controller è probabilmente il Design Pattern che più di tutti ha modificato lo schema del sistema da progettare. Il suo scopo, ovvero il problema per il quale fornisce una soluzione come tutti i Pattern, è quello di separare l'interfaccia utente dalla parte funzionale del sistema. Come suggerisce il nome, il principio del Design Pattern MVC si basa sull'individuare tre principali macroaree tematiche:

La Model che ingloba tutti i dati gestiti e la logica del programma, comprese le classi individuate nell'analisi iniziale. Questa area rappresenta il programma vero e proprio, spesso nascosto all'utente finale.

La View rappresenta l'interfaccia grafica a disposizione dell'utente, si occupa di gestire cosa deve essere visualizzato andando a reperire tali informazioni dalla macroarea Model.

Il Controller segue il principio inverso della View: registra l'input utente che viene inserito nella GUI e lo gestisce inviandolo se necessario al livello Model.

Il compito dei livelli View e Controller è quindi quello di fornire un ponte per mettere in comunicazione indiretta e controllata l'utente col sistema vero e proprio.

Vantaggi ottenuti:

Il suddividere il sistema in queste tre macroaree ha ridotto la complessità generale di progettazione.

Il testing su componenti così ben definite e separate è risultato più semplice.

L'aver un'intera macroarea Controller per la gestione dell'input e quindi la supervisione dell'interazione tra utente e sistema ha permesso di includere maggiori controlli di sicurezza per evitare malfunzionamenti voluti o involontari causati dall'utilizzo scorretto del sistema da parte dell'utente.

Come è stato applicato?

Il livello View è stato implementato semplicemente progettando [l'interfaccia grafica](#), ovvero con la creazione di file interpretabili da JavaFX come schermate da visualizzare. L'interfaccia grafica così creata sarà particolarmente reattiva e genererà eventi per qualsiasi utilizzo da parte dell'utente.

Il livello Controller è stato basato sull'omonimo Design Pattern di tipo GRASP. Viene creata una classe generale Controller che funga da radice del sistema, attraverso la quale passa ogni interazione col programma. Da questa vengono poi create delle sottoclassi Handler ognuna delle quali specializzata a gestire un particolare caso d'uso o finestra dell'applicazione. L'applicazione del Design Pattern MVC ha richiesto quindi l'utilizzo e la continua osservazione non solo del diagramma delle classi ma anche di quello Use Case.

La suddivisione delle tre principali macroaree è ben visibile nel diagramma delle classi.

3.1.2 Singleton

Il DP Singleton è stato applicato al progetto perché si è evidenziata la necessità di avere una sola istanza massima per alcune classi.

È probabilmente il Design Pattern che ha modificato di meno lo schema delle classi, dato che la sua applicazione non richiede l'aggiunta di classi di supporto ma solo l'aggiunta di alcuni metodi al loro interno.

Come è stato applicato?

Nelle classi verso le quali è stato applicato il Singleton, il metodo costruttore è stato reso privato, in modo che istanze di classi esterne non potessero usufruirne.

È stato reso disponibile e pubblico invece un metodo statico che fornisce l'istanza Singleton della classe in questione quando viene richiesto, rendendo automatica e privata la sua creazione nel caso ancora non esista.

Esempi di applicazione: il DAO che gestisce l'accesso al database è stato reso Singleton, in modo da canalizzare tutti gli accessi al DB attraverso un singolo oggetto con alti standard di sicurezza e sul quale si possono applicare precisi algoritmi di auditing. Applicato anche alla classe Responsabile Generale per il motivo descritto qui di seguito.

Vantaggi ottenuti:

A livello di logica di funzionamento, l'applicazione del Singleton ha permesso di creare una sola istanza in classi che non avevano nessun bisogno di avere più istanze. Questo permette di prevenire eventuali errori di programmazione che potrebbero portare a creare più istanze non richieste. Ad esempio, la classe Responsabile Generale che può essere vista come una persona o un ufficio di riferimento a capo di tutto in quanto a responsabilità.

A livello di sicurezza, l'avere una sola istanza permette di intensificare i controlli di sicurezza, ad esempio per l'accesso a una risorsa particolarmente sensibile come il database di voto. Inoltre la presenza di due o più entità che operano sugli stessi oggetti potrebbero creare problemi di concorrenza, ad esempio due Responsabili Generali che bloccano o modificano una votazione in corso. Per tali motivi è stato applicato il Design Pattern Singleton alla classe Responsabile Generale e alla classe che gestisce la connessione col Database.

Esempio di implementazione DP Singleton per la classe ResponsabileGenerale:

```
public class ResponsabileGenerale {  
    // Variabile che contiene l'unica istanza singleton della classe:  
    private static ResponsabileGenerale Istanza;  
    // Costruttore non accessibile da classi esterne:  
    private ResponsabileGenerale() {  
    }  
    // Metodo per ottenere l'istanza singleton  
    public static ResponsabileGenerale GetIstanza() {  
        if (Istanza == null) {Istanza = new ResponsabileGenerale} //solo nel caso l'istanza non esista ancora  
        return Istanza;  
    }  
}
```

3.1.3 DAO

Il Data Access Object è un Design Pattern architetturale che permette di astrarre l'accesso e l'utilizzo dell'entità Database, facilitando così la separazione logica tra le classi che gestiscono l'accesso al DB con quelle che ne richiedono o ne utilizzano i risultati delle interrogazioni.

Vantaggi ottenuti:

La creazione di una classe specifica DAO (implementata come Singleton come già discusso) che implementi metodi per l'accesso e l'interrogazione del DB permette di avere classi generiche del sistema (come ad esempio la classe totem) che non necessitano di implementare loro stesse del codice SQL ma sfruttano quello già presente nelle classi DAO, accessibili tramite metodi specifici. La creazione di comandi SQL è una delle situazioni potenzialmente pericolose per la sicurezza del sistema, in quanto un comando creato in maniera errata o una vulnerabilità SQL non gestita potrebbe compromettere l'integrità dei dati. Per questo motivo la classe DAO ha dei metodi predisposti alla creazione di stringhe SQL, nelle quali vanno aggiunti solo i nomi delle tabelle di destinazione e particolari filtri di ricerca. Questo minimizza la possibilità di subire attacchi di tipo SQLInjection.

Come è stato applicato?

È possibile notare nello schema delle classi come in realtà sia stata aggiunta solo una classe DAO che funge da ponte tra il sistema e il DB, e che viene utilizzata dalle classi che devono accedere ai dati contenuti nel server. Il motivo per il quale l'implementazione è così semplice e incompleta è che si è scelto di ipotizzare l'utilizzo di strumenti per la gestione dei dati e della loro persistenza esterni come per esempio ORMLite (cap [3.4](#)), che risulta essere pienamente compatibile col DB in esame e che fornisce già oggetti di tipo DAO per l'accesso e l'utilizzo del DB.

L'utilizzo di soluzioni DAO esterne sfrutta il principio del riutilizzo di codice, portando così la certezza di utilizzare strumenti ampiamente testati e supportati. Qualora nella realizzazione finale si dovesse scegliere una soluzione DB differente, sarà sufficiente scegliere un diverso tipo di servizio ORM.

3.2 Vincoli

Per la discussione dei vincoli è stata presa come esempio la classe Elettore. Tale classe era già stata parzialmente creata e correlata di vincoli JML seguendo le indicazioni dell'[assignment 3](#). La classe è stata aggiornata, il codice JML commentato e aggiunti i vincoli OCL. Visionando la classe sarà possibile notare la correlazione tra vincoli JML e OCL.

L' esempio in considerazione:

```
public class Elettore {
    // attributi
    public String Nome, Cognome, Nazione, Comune, Sesso, CF, ID;
    public int GiornoNascita, MeseNascita, AnnoNascita, età;
    private /*@ spec_public @*/ boolean Voto;

    //variabili di supporto, leggere dettagli nei commenti successivi
    //public int GiornoOggi=17, MeseOggi=11, AnnoOggi=2021;

    //@ public invariant (Sesso == "M" || Sesso == "F") ;
    //@ public invariant ((Nome != null) && (Cognome != null));
    //@ public invariant ((Nazione == "Italia") ==> (Comune != null));

    {
        context Elettore
        Inv: this.Sesso = #F or this.Sesso = #M
        Inv: this.Nome!=null and this.Cognome!=null
        Inv: this.Nazione=#Italia implies this.Comune!=null
        Inv: this.allInstances.isUnique(ID)
        context Elettore::esprimi_voto()
        Pre: this.età >= 18
    }
}
```

Si noti la relazione tra i vincoli JML, già testati e derivanti dall'assignment 3, e i nuovi vincoli OCL.

Rispetto a JML, i nuovi vincoli comprendono un controllo aggiuntivo sull'unicità degli ID, vincolo che dovrà essere presente in ogni classe che comprende un attributo ID (Votazione, Totem, Seggio).

3.3 Testing/Controllo parametri input

Per la fase di test è stato preso come esempio il testing della funzione di controllo input presente nella schermata di LoginTecnico, e più precisamente nella relativa classe di controllo LoginTecnicoController. Sono presenti due textfield nei quali teoricamente un tecnico inserisce le sue credenziali per accedere. Sono stati aggiunti dei controlli per assicurarsi che le credenziali rispettino le norme di sicurezza e che non contengano particolari stringhe SQL. Questo tipo di controllo è stato testato attraverso junit applicato a una classe esterna di test, la cui funzione è stata quella di immettere varie combinazioni di credenziali, alcune valide altre no, per vedere se i warning e i messaggi di errore visualizzati all'utente compaiono correttamente.

La copertura applicata ai casi di test è il branch coverage.

Maggiori dettagli del testing:

La classe LoginTecnicoController fornisce due metodi pubblici per gestire il controllo input utente, nello specifico i metodi controlloInputUser(string) e controlloInputPass(string). Il motivo per cui ne sono stati creati due è perché i vincoli che devono seguire il nome utente e la password sono diversi tra loro, la password deve seguire vincoli più restrittivi come ad esempio una lunghezza minima.

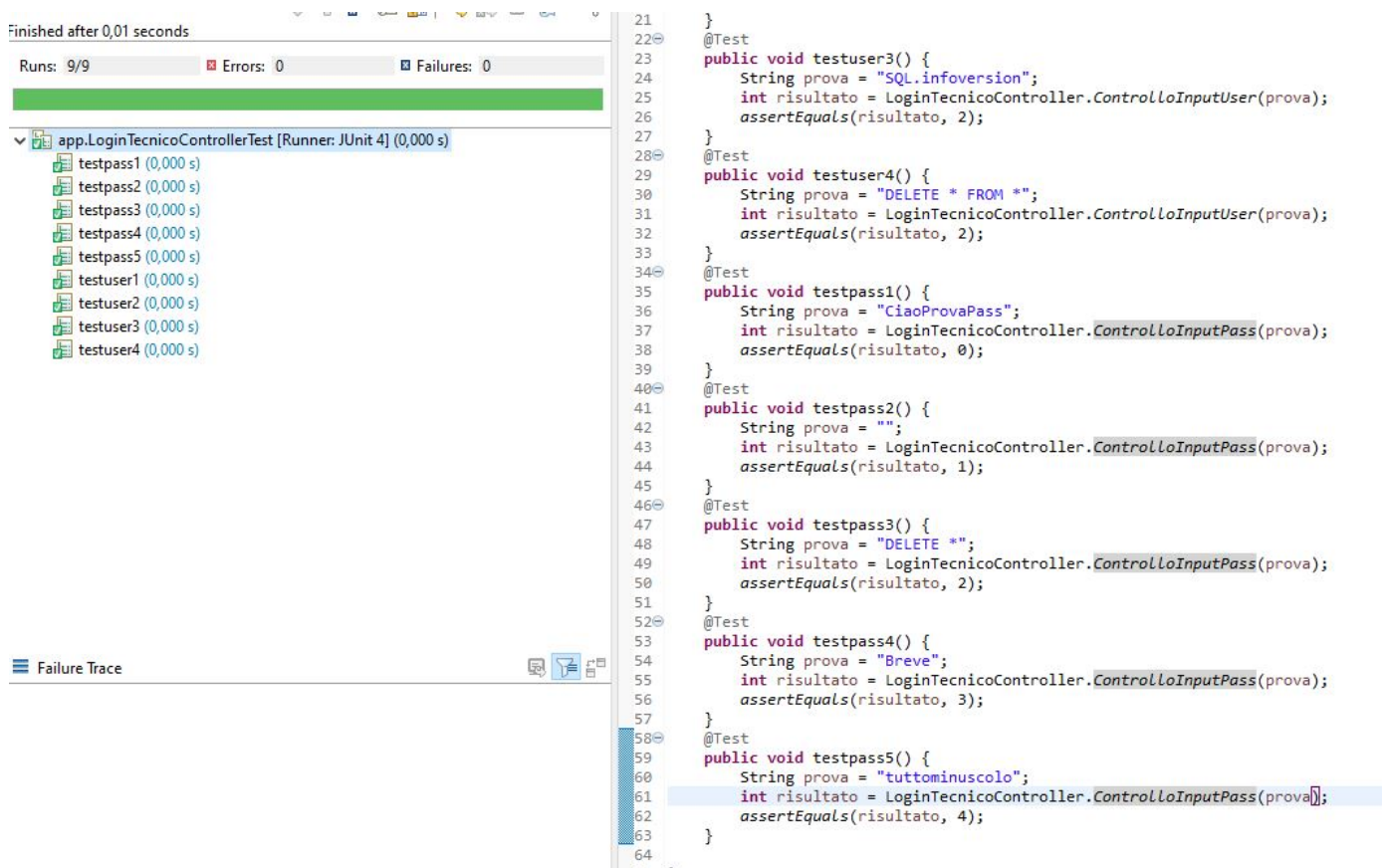
I due metodi eseguono una verifica sulla stringa che viene passata loro e in base all'esito del controllo forniscono come risultato un numero. Ad ogni numero fornito corrisponde un particolare risultato, con ad esempio 0 che corrisponde a "stringa valida", 1 che corrisponde a "textfield lasciato vuoto" oppure 2 se viene rilevato all'interno del testo alcuni comandi SQL che potrebbero far pensare a un possibile tentativo di attacco al Database.

Nell'apposita classe di test junit chiamata LoginTecnicoControllerTest, vengono testate le due funzioni fornendo loro una serie di diverse stringhe come parametro, alcune corrette, altre con errori ben specifici, e la correttezza dei metodi viene valutata confrontando il risultato del caso di test col risultato ideale atteso.

L'implementazione scelta per i metodi di controllo input non fornisce un risultato binario giusto/sbagliato ma evidenzia anche quale tipo di problema è stato riscontrato. Questo ha permesso di costruire dei testcase più completi e dettagliati.

L'esito del test `testuser2()` inizialmente ha dato esito negativo e ha permesso di individuare un bug nel riconoscere la differenza tra stringa vuota e stringa null;

Qui di seguito alcune schermate che mostrano i test eseguiti con JUnit e l'esempio di funzionamento del controllo parametri input:



```
21 }
22 @Test
23 public void testuser3() {
24     String prova = "SQL.infoversion";
25     int risultato = LoginTecnicoController.ControlloInputUser(prova);
26     assertEquals(risultato, 2);
27 }
28 @Test
29 public void testuser4() {
30     String prova = "DELETE * FROM *";
31     int risultato = LoginTecnicoController.ControlloInputUser(prova);
32     assertEquals(risultato, 2);
33 }
34 @Test
35 public void testpass1() {
36     String prova = "CiaoProvaPass";
37     int risultato = LoginTecnicoController.ControlloInputPass(prova);
38     assertEquals(risultato, 0);
39 }
40 @Test
41 public void testpass2() {
42     String prova = "";
43     int risultato = LoginTecnicoController.ControlloInputPass(prova);
44     assertEquals(risultato, 1);
45 }
46 @Test
47 public void testpass3() {
48     String prova = "DELETE *";
49     int risultato = LoginTecnicoController.ControlloInputPass(prova);
50     assertEquals(risultato, 2);
51 }
52 @Test
53 public void testpass4() {
54     String prova = "Breve";
55     int risultato = LoginTecnicoController.ControlloInputPass(prova);
56     assertEquals(risultato, 3);
57 }
58 @Test
59 public void testpass5() {
60     String prova = "tuttominuscolo";
61     int risultato = LoginTecnicoController.ControlloInputPass(prova);
62     assertEquals(risultato, 4);
63 }
64 }
```



```

void pressione(ActionEvent event) {

    //lettura credenziali immesse
    String user =username.getText().toString();
    String pass = String.valueOf(password.getText());
    //controlloInput su username e password, con relativo messaggio di errore
    int UserValido = ContolloInputUser(user);
    if(UserValido == 1) {labelavvisouser.setText("Username non può essere vuoto");}
    if(UserValido == 2) {labelavvisouser.setText("Possibile tentativo SQLInjection");}
    int PassValido = ContolloInputPass(pass);
    if(PassValido == 1) {labelavvisopass.setText("Password non può essere vuoto");}
    if(PassValido == 2) {labelavvisopass.setText("Possibile tentativo SQLInjection");}
    if(PassValido == 3) {labelavvisopass.setText("La password deve contenere almeno 8 caratteri");}
    if(PassValido == 4) {labelavvisopass.setText("La password deve contenere almeno una maiuscola");}

}

@FXML
void initialize() {
    assert button != null : "fx:id=\"button\" was not injected: check your FXML file 'LoginTecnico.fxml'";
    assert labelavvisopass != null : "fx:id=\"labelavvisopass\" was not injected: check your FXML file 'LoginTecnico.fxml'";
    assert labelavvisouser != null : "fx:id=\"labelavvisouser\" was not injected: check your FXML file 'LoginTecnico.fxml'";
    assert password != null : "fx:id=\"password\" was not injected: check your FXML file 'LoginTecnico.fxml'";
    assert username != null : "fx:id=\"username\" was not injected: check your FXML file 'LoginTecnico.fxml'";
}

public static int ContolloInputUser(String testo) {
    if (testo.isEmpty()) {return 1;}
    if (testo.contains("SQL")||testo.contains("SELECT")||testo.contains("DELETE")) {return 2;}
    return 0;
}

public static int ContolloInputPass(String testo) {
    if (testo.isEmpty()) {return 1;}
    if (testo.contains("SQL")||testo.contains("SELECT")||testo.contains("DELETE")) {return 2;}
    if (testo.length()<8) {return 3;}
    String minuscolo = testo.toLowerCase();
    if (minuscolo.equals(testo)) {return 4;}
    return 0;
}

```

Finestra di login con risultati del controllo input visibili nel cap [3.6.1](#)

3.4 Gestione dati persistenti

Come già accennato in [3.1.3](#), la connessione al Database avviene tramite DAO. Come DBMS si ipotizza l'uso di SQLite.

L'accesso e la gestione dei dati persistenti è stato delegato a ORMLite.

Dal sito del produttore:

Object Relational Mapping Lite (ORM Lite) provides some simple, lightweight functionality for persisting Java objects to SQL databases while avoiding the complexity and overhead of more standard ORM packages.

L'utilizzo di ORMLite sfrutta il principio di riutilizzo del codice, permettendo di usare codice già ampiamente testato e particolarmente sicuro al quale viene delegata la funzione di accesso al DB e la gestione dei dati persistenti. Tra le funzioni presenti, le più importanti sono:

- Fornite classi astratte di tipo DAO per l'accesso al Database.
- Funzionalità di QueryBuild particolarmente flessibile, per la creazione guidata di query SQL. Tale funzione minimizza la possibilità di attacchi SQLInjection.
- Funzioni di supporto per query "ripetitive" e usate di frequente, quali possono essere il Totem che per ogni accesso chiede di reperire le votazioni attive e disponibili.
- Creazione automatica di query tramite API java.
- Riesce ad individuare in automatico le tabelle e le loro relazioni all'interno del DB.

L'ORM scelto inoltre supporta tutti i principali sistemi DBMS, perciò l'ipotetica sostituzione di SQLite con altri DBManager rimarrà circoscritta solo al server.

Maggiori info sul prodotto: <https://ormlite.com/>

Per quanto riguarda i Database presenti sul server, si ipotizzano queste due configurazioni, entrambe caricate e gestite dallo stesso DBManager.

Database server Login:

Due tabelle, una chiamata "Tecnico" che contiene tutte le informazioni e i dati personali e lavorativi dei tecnici certificati, e una chiamata "Credenziali" contenete username e password (codificate). Le due tabelle sono collegate con una relazione 1 a 1, e in caso di richiesta di login da parte di un totem viene fatta una SELECT nella tabella Credenziali. Se tale interrogazione fornisce un risultato, si procede a reperire le info del corrispondente Tecnico nella tabella Tecnico e si verifica se l'utente in questione risulti ancora in servizio e abilitato ad accedere. In caso positivo, si fornisce la conferma di login.

Database server Voto:

Una tabella chiamata "Votazioni" contiene tutte le votazioni inserite nel server, comprese quelle già chiuse. Per ogni votazione sono disponibili diversi attributi che

ne indicano la tipologia, il quorum, ecc. Questa è la tabella verso la quale il Responsabile Generale agisce direttamente per inserire o modificare una votazione.

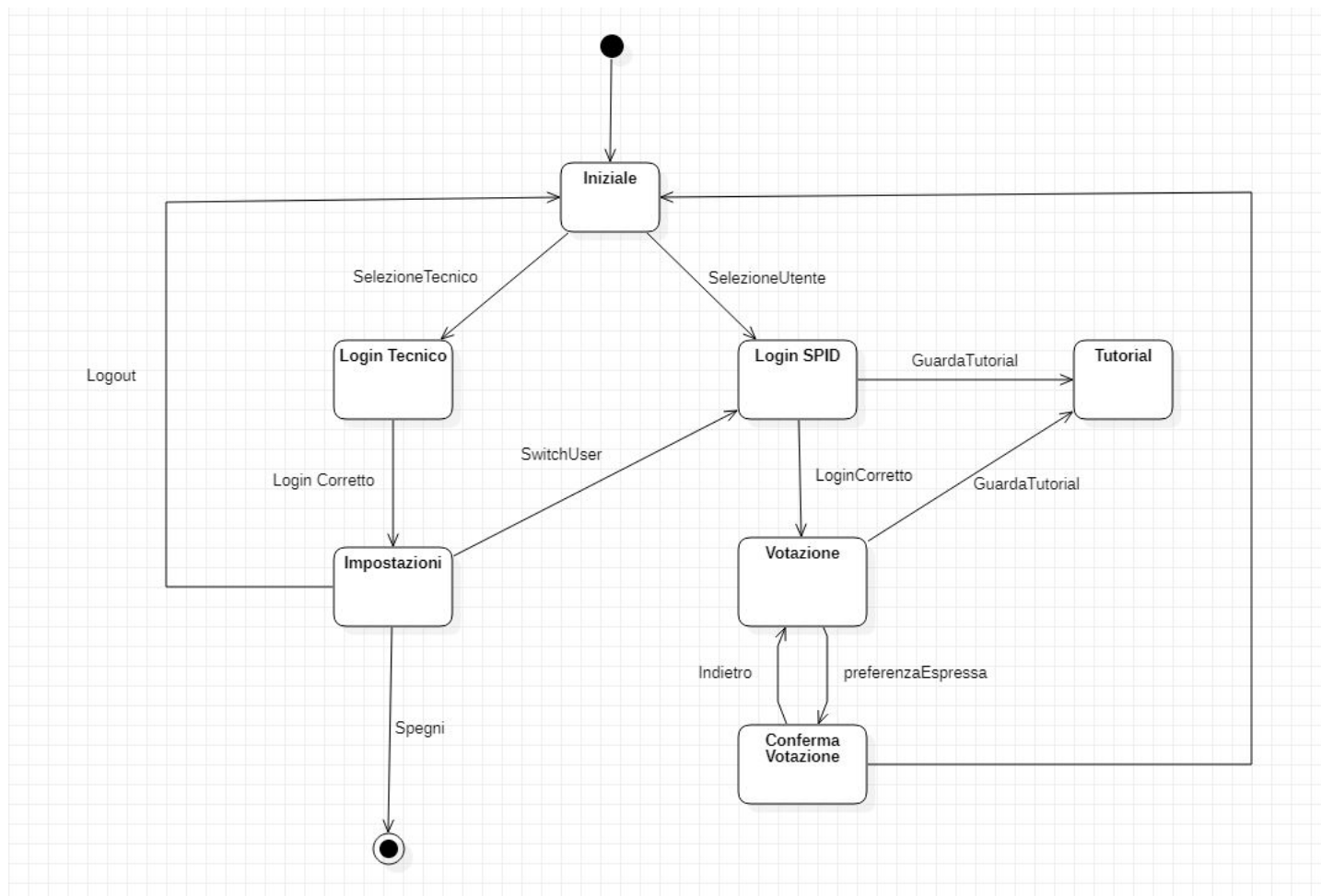
Il totem dopo il login elettore, consulta tale tabella, seleziona le votazioni al momento aperte e verifica attraverso i dati anagrafici dell'elettore se può esprimere il voto per quelle votazioni (si pensi ad esempio a elezioni comunali visibili solo a utenti con residenza registrata presso quel comune).

Vi è inoltre una tabella chiamata "Voti" contenente un record per ogni preferenza espressa. Ogni record contiene un id generato a random per identificare un elettore, l'id della votazione di riferimento, e la preferenza espressa.

Tale tabella viene poi consultata al termine della votazione per fornire i risultati.

Nella tabella Voti l'id dell'elettore viene generato a random e non è possibile risalire all'elettore in questione, per soddisfare il requisito di anonimato. Perché questo funzioni, si suppone che il controllo dell'esistenza di tale utente avvenga mediante consultazione degli archivi dello Stato durante la fase di login con SPID.

3.5 Navigation map



La mappa di navigazione rappresenta le possibili combinazioni di schermate che si possono alternare sul totem.

Si noti come tutte le schermate indicate come se fossero degli stati hanno lo stesso nome della corrispettiva schermata in formato .fxml, visibili nel capitolo [3.6](#).

La descrizione delle transizioni tra due schermate indicata nella mappa può rappresentare l'input utente, come il click sul pulsante Indietro, o un evento da parte del sistema, come ricevere dal server la conferma di login corretto.

3.6 Schermate di esempio

Schermata “Iniziale” con scelta tipologia di utente




Schermata “LoginSPID” con accesso QR (come tutti i servizi statali con SPID)





Schermata “Votazione” che visualizza solo le elezioni per le quali si ha diritto

Votazione.fxml

Opzioni


Assistenza


Tutorial


Switch user

Indietro

Logout

Voti disponibili:

Voto ordinale: Non Disponibile


Voto categorico: Non Disponibile


Referendum: Disponibile

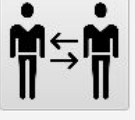
Schermata “VotazioneConferma” che mostra la possibilità di voto con scheda bianca

VotazioneConferma.fxml

Opzioni


Assistenza


Tutorial


Switch user

Indietro

Logout

Confermare voto espresso

Voto ordinale: Non Disponibile



Voto categorico: Non Disponibile

Referendum: ☐ SI ☒ NO ☐ Nessun voto

Conferma e Esci

Schermata “LoginTecnico” che integra i [controlli sull'input](#) nella classe controller

LoginTecnico.fxml

Opzioni

Assistenza

Tutorial

Indietro

Inserire credenziali d'accesso

USERNAME

PASSWORD

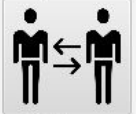

ACCEDI

[Password dimenticata?](#)

Schermata “Impostazioni” accessibile solo da un Tecnico Certificato

Impostazioni.fxml

Info sistema
Connessione: **OK**
Periodo di attività:
Votazioni attive:

Opzioni

Switch user

Spegni totem

Logout

Avvia diagnostica

Test connessione

Visualizza LOG

Data ultima diagnostica:

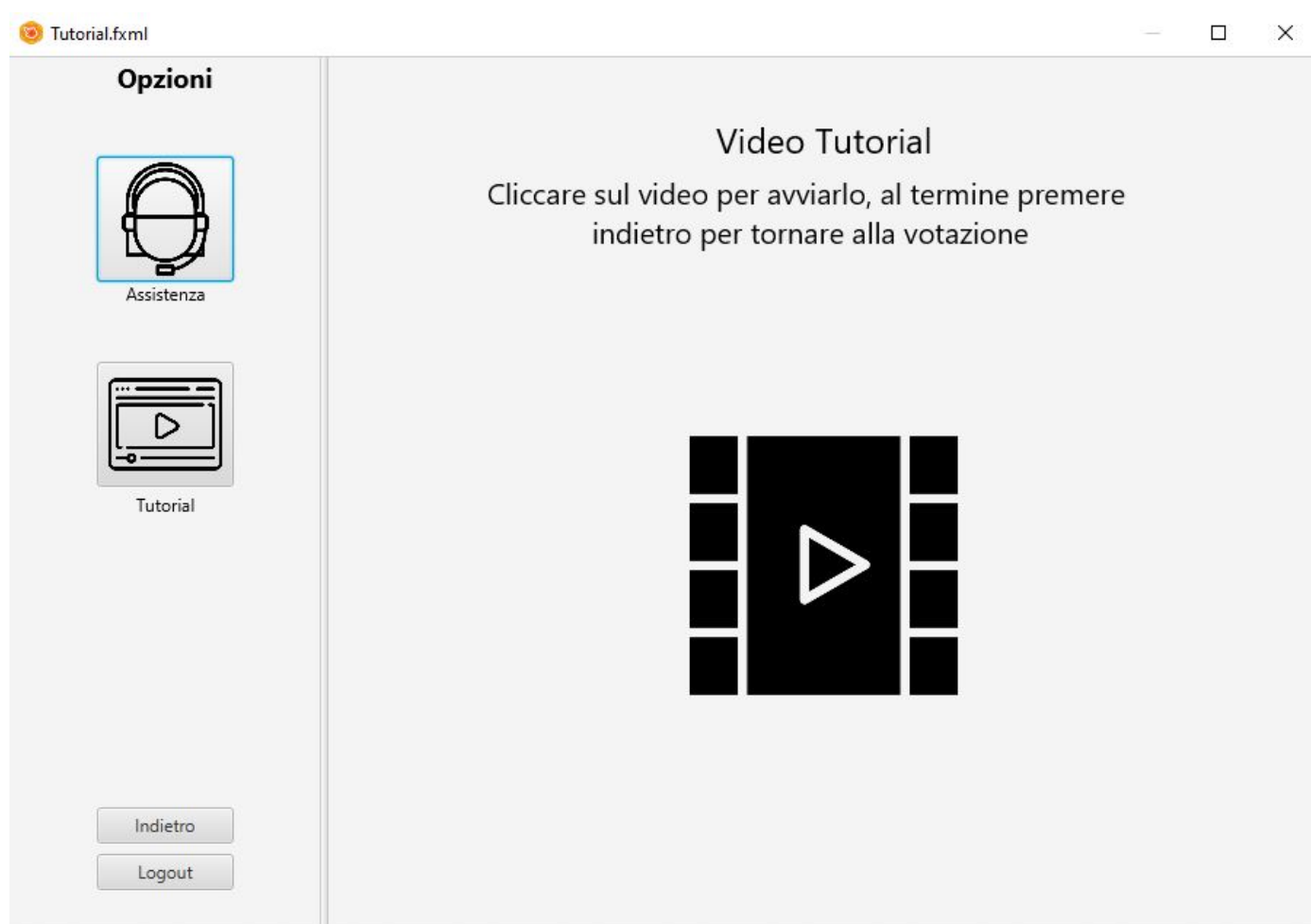
Esito:

Connessione:

Qualità della rete:

Invia LOG


Schermata "Tutorial" accessibile tramite menù a lato per gli utenti




3.6.1 Esempi controllo input

LoginTecnico.fxml

Opzioni


Assistenza


Tutorial

Indietro

Inserire credenziali d'accesso

USERNAME

Username non può essere vuoto


PASSWORD


ACCEDI

[Password dimenticata?](#)

LoginTecnico.fxml

Opzioni


Assistenza


Tutorial

Indietro

Inserire credenziali d'accesso

USERNAME

Possibile tentativo SQLInjection

PASSWORD

ACCEDI

[Password dimenticata?](#)

Opzioni

Assistenza



Tutorial

Indietro

Inserire credenziali d'accessoUSERNAME PASSWORD

La password deve contenere almeno 8 caratteri

ACCEDI

[Password dimenticata?](#)**Opzioni**

Assistenza



Tutorial

Indietro

Inserire credenziali d'accessoUSERNAME PASSWORD

La password deve contenere almeno un carattere maiuscolo

ACCEDI

[Password dimenticata?](#)

Il controllo input è implementato all'interno della classe controller della schermata di login per i tecnici certificati

Ai fini della valutazione finale, aggiungo le valutazioni ricevute finora:

MOD 1 appello 21 Febbraio 2022, voto 26.

MOD 2 appello 13 Giugno 2022, voto 27.

Valutazione positiva dei cinque assignment.

Rimango in attesa della valutazione finale del corso.

Grazie,

Lorenzo Gardelli, matricola 907502 , data di consegna 19 Luglio 2022.