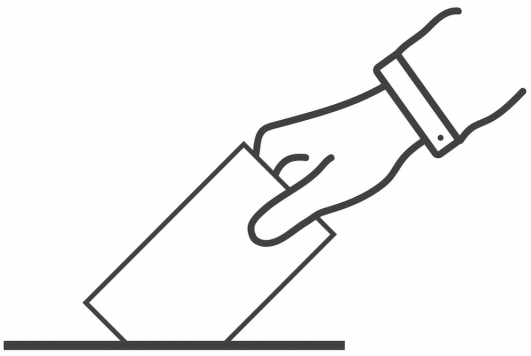
easyVote

Jun Xiang Federico Zhou M: 941519 — Chen Sheng Angelo M: 942777

June 15, 2022



easyVote

Contents

1 Introduction

Democracy is a governmental model based of the belief that the population, or all eligible member of a state shall express their political intentions directly, or by electing representatives. easyVote is the new, efficient to use, comprehensive and dynamic electronic voting and balloting system. With easyVote governmental (and non) entities can expect to use an easy to comprehend system to manage electoral systems.

1.1 Objectives:

This RASD/SRS document shall include a glossary, general description, alongside a comprehensive specification of functional and non-function descriptions, use case diagrams, class diagrams, sequence diagrams, components diagrams, used design patterns section, database description, specification and validations with JML, user interface description section, testing section and installation and settings section.

1.2 User Stories

- As a voter in Milan, I want to be able to vote in the regional election at a distance so that I can ensure my and the other voters' safety.
- As a governmental entity in Italy, I want to be able offer my citizens the possibility to vote exclusively at a distance to mitigate the effects of the pandemic.
- As a voter in the European Union, I want to be able to vote in the European election even while abroad.
- As a voter in Argentina, I want to be able to vote in a national referendum without having to deal with an overly complicated UI.

1.3 Terminology

- easyVote: is a electronic voting system, for governmental and non governmental entities able to manage different voting systems
- representatives: a person selected in an election to represent the political intention of the population
- voter: a singular portion of the eligible voter population
- electronic voting and balloting system: a voting and balloting system in which functions are primarily delegated to computers or computer infrastructure

2 Requirements Specification

easyVote is an electronic voting system, for governmental and non governmental entities able to manage different voting systems. Using easyVote eligible voters are able to vote in presence, or at a distance. Different voting system include and are not limited to

- ordinary vote: in which the voter is asked to order the candidates based on their preference
- categorical vote: in which the voter is asked to select a preferred candidate among the candidates based on their preference
- referendum: in which the voter is asked to express agreement or disagreement with the subject of the referendum

When the voting process is complete, the result of the election is determined based on:

- simple majority: the winning candidate is the one which received the majority of the votes
- absolute majority: the winning candidate is the one which received 50% + 1 votes
- referendum without quorum: the votes are counted independently of the showing of the voter base
- referendum with quorum: the votes are counted if and only if the majority of the eligible voter base voted

easyVote users shall be divided into voters and administrators.

2.1 Voters/users

Voters must be able to vote in presence, or at a distance, using the easyVote portal, after a proper identification process, different based on the type of vote casted.
If the vote is casted in presence, the identification process happens by requesting an identification document, if the vote is casted at a distance the identification process is delegated to the easyVote system.
Each voter is eligible to vote once, the vote is secret and cannot be reconnected to the caster.

2.2 Administrators

Administrator are special users able to configure new vote sessions. Vote session are characterised by voting mode, voting session type, which include and are not limited to: ordinary voting session, categorical voting session, referendum voting session.

Administrator are also able to specify the candidates, set the start and the end of the election process, as well as displaying the winning candidate and/or the winning candidates.

2.3 Functional Requirements

- The easyVote system shall be clear, comprehensible and easy to use by every voter. Special measures shall be implemented to support voters with special conditions
- Voter users shall register themselves through a dedicated easyVote portal
- Administrator users shall be authenticated using a dedicated easyVote portal
- Administrator shall be able to create, manage, modify and display the results of an election
- The easyVote system shall contain a manual to explain the different scrutinizing criteria
- Ballots shall include images, text, icons, symbols to ease the recognition of the preferred candidate by a voter
- The easyVote system shall recognise blank votes.
- Ballots shall be recognised by the system if and only if the voter has specified a valid preference, or has expressed no preference at all (blank vote)
- After the vote is casted a confirmation window shall be displayed to allow the voter to verify the correctness of the vote
- A voter shall be able to verify his vote after the vote is casted
- Users shall be able to resume a previously started voting sessions if the voting window has not closed yet
- The easyVote system shall register all entities with a UID.
- Administrators shall be able to create sample ballots for informative purposes.
- The easyVote system shall provide appropriate information on the ballot for each candidate

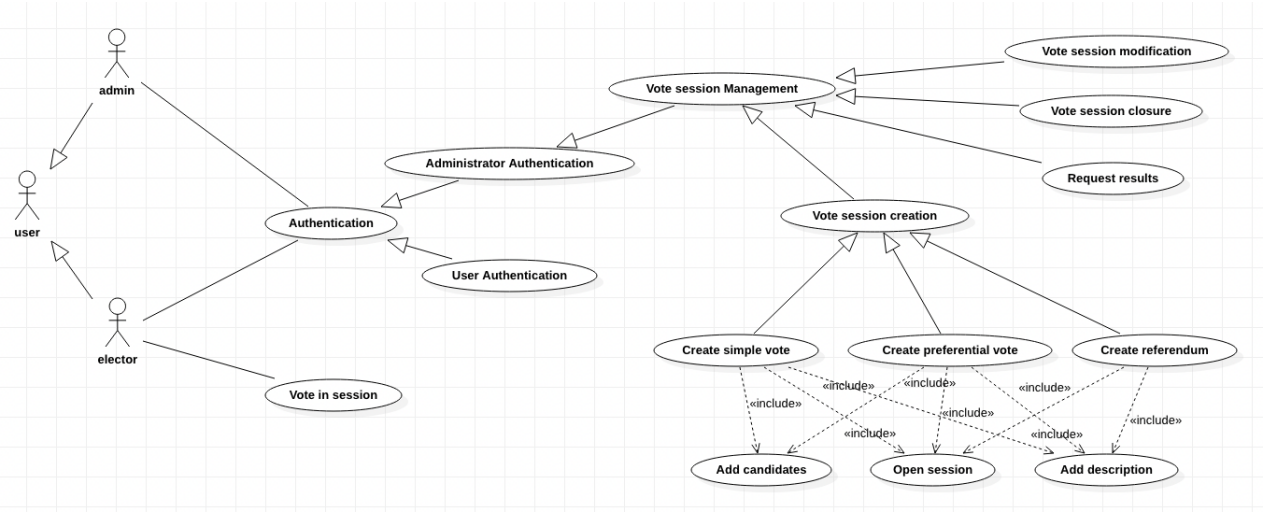
2.4 Non-Functional Requirements

- The code shall be commented according to "JAVADOC" guidelines;
- The easyVote online portal shall only use the HTTPS protocol and never the HTTP protocol;
- The drive on which the the easyVote system is installed shall have a level 6 RAID level for extra redundancy;
- The easyVote portal shall be accessible through a web application light enough to be run on any digital device equipped with an Internet connection;
- The easyVote system shall be impartial in all stages of use, the system shall never in any way influence the voter directly or indirectly;
- The easyVote system shall guarantee a high degree of clearness and ease of use through the use of icons, symbols, images;
- The easyVote system shall manage users and administrators information according to the GDPR guidelines.

2.5 Security Requirements

- If a voter is able to vote through multiple modes, the system shall recognise only the last vote casted;
- A voter, to vote at a distance, shall identify himself with a high security protocol. The identification process shall be delegated to outside entities such as SPID;
- The easyVote system shall never expose information regarding the voter, a vote casted, the administrators. This principle shall be respect during all stages use.

3 Use case diagrams



- Authentication:
  a use case accessed by admins and electors alike to authenticate themselves using their personal username and password selected during the registration process.
  The system keeps track of the authenticated user during the use of the program;

- Vote in session:
  a use case accessed by elector to vote in open session. The system automatically tracks and removes sessions in which the user has already voted in. The user can only accessed open voting sessions;

- Vote session management:
  a use case accessed by admins to manage voting session. By "manage voting session" we mean: vote session modification, vote session closure, vote session results request, vote session creation;

- Vote session modifications:
  a use case accessed by admins to add modifications to open vote sessions. Modification include and are not limited to: adding and removing candidates, changing a vote description;

- Vote session closure:
  a use case accessed by admins to close an open voting session;

- Request results:
  a use case accessed by admins to request the result of a closed voting session;

- Vote session creation:
  a use case accessed by admins to request the opening of a new voting session. During the creation process the admin can define the following parameters and more: description, candidates, type of voting session;

- Create categorical vote:
  a use case accessed by admins to create a simple voting session. A simple voting session is a voting session in which the winning strategy is simple majority;

- Create ordinary vote:
  a use case accessed by admins to create a ordinary voting session. An ordinary voting session is a voting session in which a user is asked to select its favourite candidates in a descending order;

- Create referendum:
  a use case accessed by admins to create a referendum session. A referendum voting session is a voting session in which a user is asked to "in favour" or "not in favour" or leave blank ballot to a requested referendum change.

3.1 Scenery descriptions

In this section we look into the scenery description for the following actions:

- Login
- Registration
- castVote
- Create voting session
- Modify voting session
- Vote in a referendum
- Request results

Nome	Login
Scopo	Permettere all'utente di autenticarsi presso il sistema easyVote
Attore/i	Utente
Pre-Condizioni	
Trigger	Click del bottone “Login” nella schermata “Home” del sistema easyVote
Descrizione sequenza eventi	1. Il sistema verifica se è il cliente è autenticato al sistema nel momento dell’innesco del caso d’uso 2. Se il cliente non è autenticato, il sistema fornisce un form comprendente i campi nome utente e password 3. Il cliente compila i campi richiesti e invia il form 4. Il sistema verifica che i dati inseriti siano corretti ed autentica il cliente
Alternativa/e	2a. Se Il cliente già autenticato, il caso d’uso termina
Post-Condizioni	L’utente è autenticato

Nome	Registration
Scopo	Permettere all'utente di registrarsi presso il sistema easyVote
Attore/i	Utente
Pre-Condizioni	
Trigger	Click del bottone “Registrati” nella schermata “Home” del sistema easyVote.
Descrizione sequenza eventi	1. Il cliente inserisce in un apposito form i dati anagrafici e altri dati, come e-mail e password, richiesti dal sistema 2. Il sistema restituisce un messaggio di avvenuta registrazione 2a. Se il cliente è già esistente, il sistema restituisce un errore e il caso d’uso termina
Alternativa/e	
Post-Condizioni	L'utente è registrato

Nome	castVote
Scopo	Permettere al voter di votare per il candidato, i candidati o il partito da lui preferito
Attore/i	Elettore
Pre-Condizioni	il Voter deve essere autenticato
Trigger	Click del bottone “Vota” nella schermata “Home” del sistema easyVote
Descrizione sequenza eventi	1. All’elettore viene chiesto il voting session nella quale si ha intenzione di votare 2. L’elettore effettua la scelta 3. A seconda del tipo di votazione della voting session, il sistema chiede all’elettore la sua preferenza / le sue preferenze. 4. Il sistema chiede all’elettore di confermare le preferenze. 5. Il sistema restituisce un messaggio di avvenuto voto.
Alternativa/e	4a. Se l’elettore non esprime alcuna preferenza o non esprime un numero di preferenze pari a quelle richieste dalla voting session il sistema restituisce un errore e il caso d’uso termina.
Post-Condizioni	L’elettore ha espresso il suo voto

Nome	Create voting session
Scopo	Permettere all'administrator di aggiungere nuove voting session
Attore/i	Administrator
Pre-Condizioni	L'administrator deve essere autenticato
Trigger	Click del bottone “modify voting session” nel sistema easyVote.
Descrizione sequenza eventi	1. L'admin seleziona il tipo di voting session 2. L’admin seleziona il criterio di vittoria del voting session 3. L’admin inserisce i candidati alla voting sessione 4. L’admin seleziona i criteri degli elettori 5. Il sistema restituisce un form di conferma
Alternativa/e	
Post-Condizioni	E' creata una nuova voting session

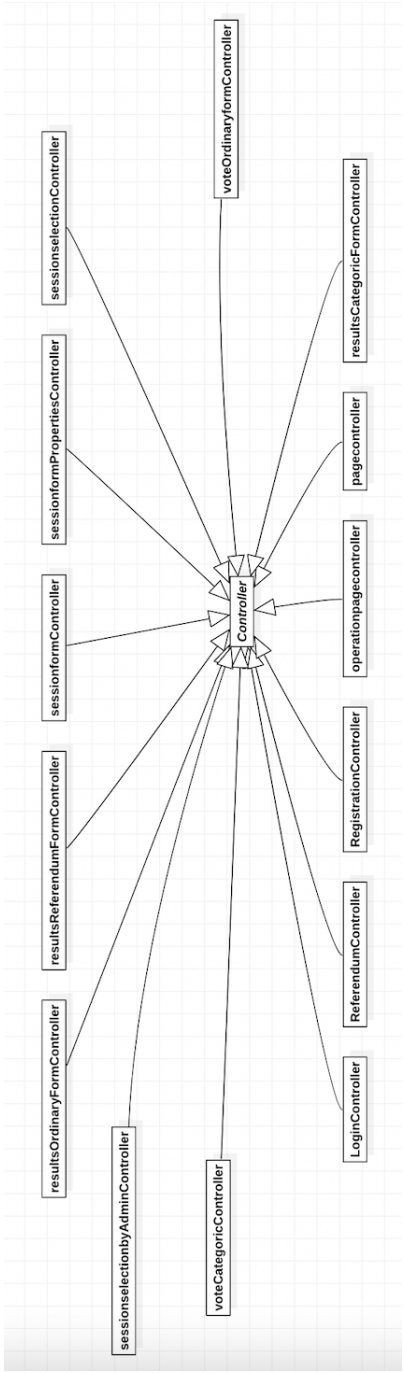
Nome	Modify voting session
Scopo	Permettere all’administrator di modificare una voting session
Attore/i	Administrator
Pre-Condizioni	L’administrator deve essere autenticato
Trigger	
Descrizione sequenza eventi	1. L'admin viene autenticato 2. L’admin seleziona la voting session che vuole modificare 3.a L’admin apporta le modifiche richieste 4. L’admin conferma le modifiche richieste 5.a Il sistema restituisce una form di conferma
Alternativa/e	5.b Il sistema restituisce una form di errore
Post-Condizioni	La voting session è modificata

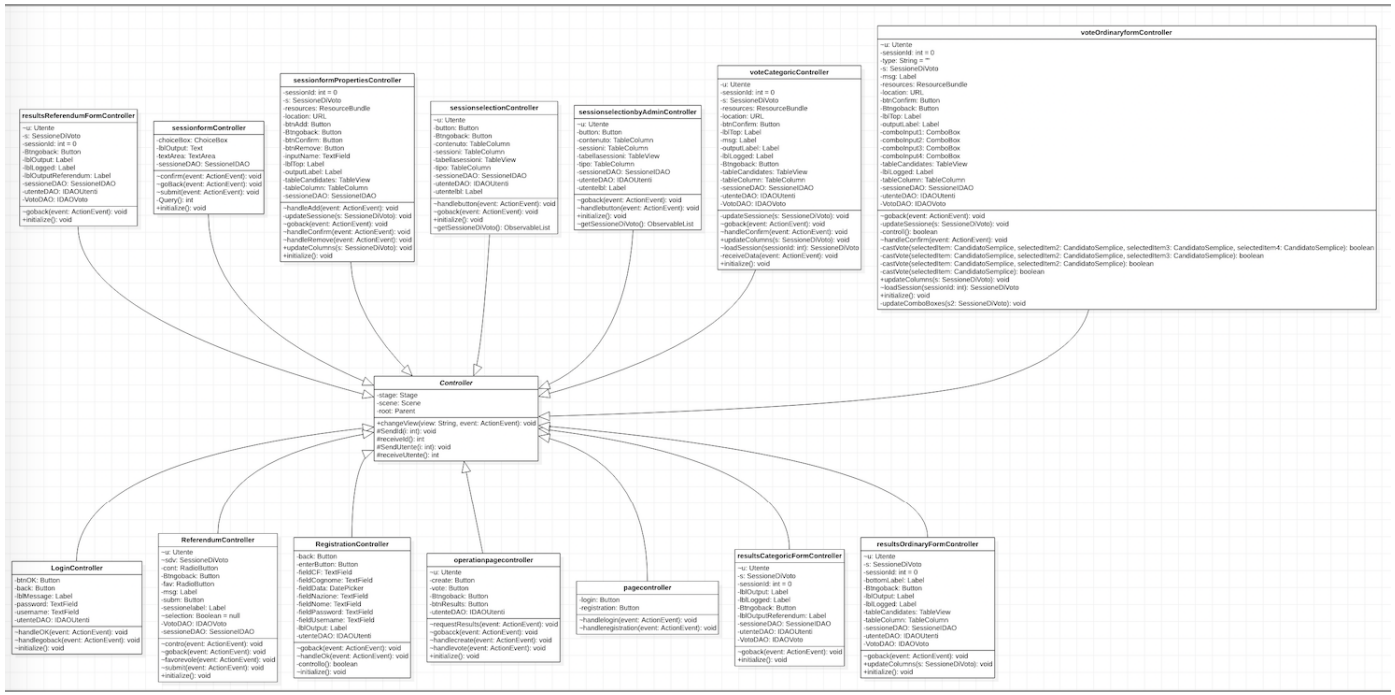
Nome	Vote in a referendum
Scopo	Permettere allo user di votare in un referendum
Attore/i	Elector
Pre-Condizioni	L’elector deve essere autenticato
Trigger	Click del bottone “vote” dopo l’autenticazione
Descrizione sequenza eventi	1. Lo user viene autenticato 2. Lo user seleziona il referendum in cui vuole votare 3. Lo user seleziona “in favore”, “contro” o lascia ballot bianco 4. Il sistema restituisce una form di conferma 5. Il sistema registra il voto, e rimuove la possibilità di votare nuovamente
Alternativa/e	
Post-Condizioni	L’elector ha votato in un referendum

Nome	Request results
Scopo	Richiedere al sistema di visualizzare il risultato di una voting session chiusa da parte dell'admin
Attore/i	Admininistrator
Pre-Condizioni	L'administrator è autenticato
Trigger	Click del bottone "chiudi voting session"
Descrizione sequenza eventi	1. L'admin viene autenticato 2. L'admin clicca il bottone "chiudi voting session" 3. Il sistema chiede di selezionare la voting session da chiudere 4. Il sistema restituisce una form di visualizzazione
Alternativa/e	
Post-Condizioni	La voting session risulta chiusa

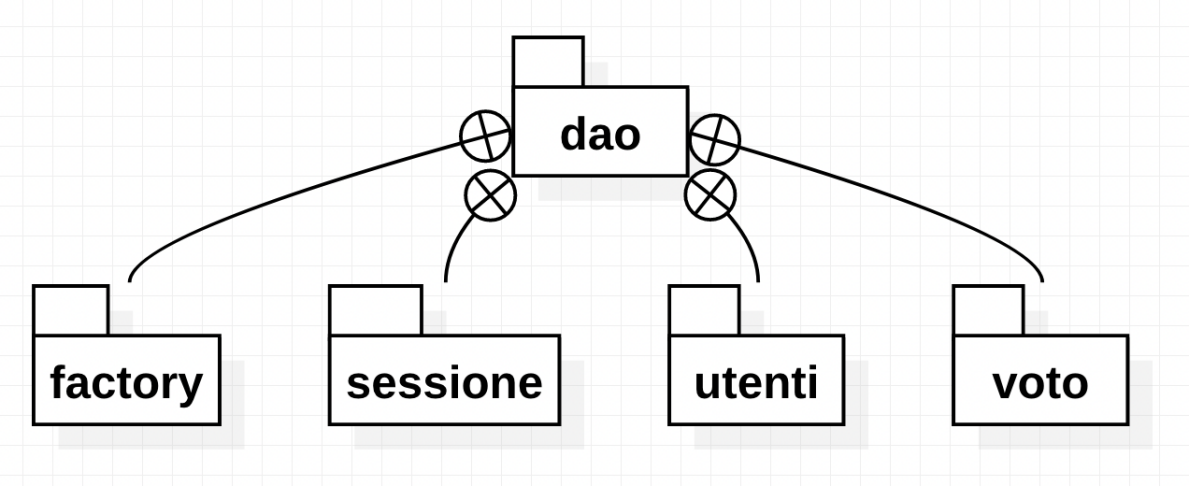
4 Class diagrams

4.0.1 Controllers

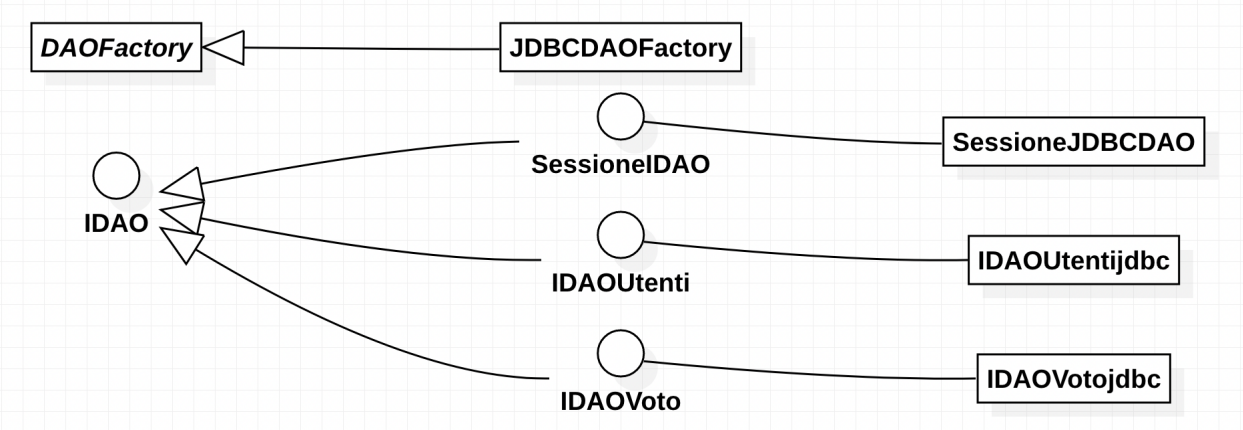



resultsReferendumFormController
-u: Utente
-s: SessioneDiVoto
-sessionId: int = 0
-Btngoback: Button
-lblOutput: Label
-lblLogged: Label
-lblOutpudReferendum: Label
-sessioneDAO: SessioneIDAO
-utenteDAO: IDAOUtenti
-VotoDAO: IDAOVoto
-goback(event: ActionEvent): void
+initialize(): void
sessionformController
-choicelBox: ChoiceBox
-lblOutput: Text
-textArea: TextArea
-sessioneDAO: SessioneIDAO
~confirm(event: ActionEvent): void
~goback(event: ActionEvent): void
~submit(event: ActionEvent): void
-Query(): int
+initialize(): void
sessionformPropertiesController
-sessionId: int = 0
-s: SessioneDiVoto
-resources: ResourceBundle
-location: URL
-btnAdd: Button
-Btngoback: Button
-btnConfirm: Button
-btnRemove: Button
-inputName: TextField
-lblTop: Label
-outputLabel: Label
-tableCandidates: TableView
-tableColumn: TableColumn
-sessioneDAO: SessioneIDAO
~handleAdd(event: ActionEvent): void
-updateSessione(s: SessioneDiVoto): void
-goback(event: ActionEvent): void
-handleConfirm(event: ActionEvent): void
-handleRemove(event: ActionEvent): void
+updateColumns(s: SessioneDiVoto): void
+initialize(): void
sessionselectionController
-u: Utente
-button: Button
-Btngoback: Button
-contenuto: TableColumn
-sessioni: TableColumn
-tabellasessioni: TableView
-tipo: TableColumn
-sessioneDAO: SessioneIDAO
-utenteDAO: IDAOUtenti
-utenteIbl: Label
~handlebutton(event: ActionEvent): void
-goback(event: ActionEvent): void
+initialize(): void
-getSessioneDiVoto(): ObservableList
sessionselectionbyAdminController
-u: Utente
-button: Button
-contenuto: TableColumn
-sessioni: TableColumn
-tabellasessioni: TableView
-tipo: TableColumn
-sessioneDAO: SessioneIDAO
-utenteDAO: IDAOUtenti
-utenteIbl: Label
-goback(event: ActionEvent): void
~handlebutton(event: ActionEvent): void
+initialize(): void
-getSessioneDiVoto(): ObservableList
voteCategoricController
-u: Utente
-sessionId: int = 0
-s: SessioneDiVoto
-resources: ResourceBundle
-location: URL
-btnConfirm: Button
-lblTop: Label
-msg: Label
-outputLabel: Label
-lblLogged: Label
-Btngoback: Button
-tableCandidates: TableView
-tableColumn: TableColumn
-sessioneDAO: SessioneIDAO
-utenteDAO: IDAOUtenti
-VotoDAO: IDAOVoto
-updateSessione(s: SessioneDiVoto): void
-goback(event: ActionEvent): void
~handleConfirm(event: ActionEvent): void
+updateColumns(s: SessioneDiVoto): void
-loadSession(sessionId: int): SessioneDiVoto
-receiveData(event: ActionEvent): void
+initialize(): void
voteOrdinaryformController
-u: Utente
-sessionId: int = 0
-type: String = ""
-s: SessioneDiVoto
-msg: Label
-resources: ResourceBundle
-location: URL
-btnConfirm: Button
-Btngoback: Button
-lblTop: Label
-outputLabel: Label
-comboInput1: ComboBox
-comboInput2: ComboBox
-comboInput3: ComboBox
-comboInput4: ComboBox
-tableCandidates: TableView
-lblLogged: Label
-tableColumn: TableColumn
-sessioneDAO: SessioneIDAO
-utenteDAO: IDAOUtenti
-VotoDAO: IDAOVoto
-goback(event: ActionEvent): void
-updateSessione(s: SessioneDiVoto): void
-control(): boolean
~handleConfirm(event: ActionEvent): void
-castVote(selectedItem: CandidatoSemplice, selectedItem2: CandidatoSemplice, selectedItem3: CandidatoSemplice, selectedItem4: CandidatoSemplice): boolean
-castVote(selectedItem: CandidatoSemplice, selectedItem2: CandidatoSemplice, selectedItem3: CandidatoSemplice): boolean
-castVote(selectedItem: CandidatoSemplice, selectedItem2: CandidatoSemplice): boolean
-castVote(selectedItem: CandidatoSemplice): boolean
+updateColumns(s: SessioneDiVoto): void
~loadSession(sessionId: int): SessioneDiVoto
+initialize(): void
-updateComboBoxes(s2: SessioneDiVoto): void
Controller
-stage: Stage
-scene: Scene
-root: Parent
+changeView(view: String, event: ActionEvent): void
#SendId(i: int): void
#received(i): int
#SendUtente(i: int): void
#receiveUtente(): int
LoginController
-btnOK: Button
-back: Button
-lblMessage: Label
-password: TextField
-username: TextField
-utenteDAO: IDAOUtenti
~handleOk(event: ActionEvent): void
~handlegoback(event: ActionEvent): void
~initialize(): void
ReferendumController
-u: Utente
-sdv: SessioneDiVoto
-cont: RadioButton
-Btngoback: Button
-fav: RadioButton
-msg: Label
-subm: Button
-sessionelabel: Label
-selection: Boolean = null
-VotoDAO: IDAOVoto
-sessioneDAO: SessioneIDAO
~contro(event: ActionEvent): void
~goback(event: ActionEvent): void
-favorevole(event: ActionEvent): void
~submi(event: ActionEvent): void
+initialize(): void
RegistrationController
-back: Button
-enterButton: Button
-fieldCF: TextField
-fieldCognome: TextField
-fieldData: DatePicker
-fieldNazione: TextField
-fieldNome: TextField
-fieldPassword: TextField
-fieldUsername: TextField
-lblOutput: Label
-utenteDAO: IDAOUtenti
~goback(event: ActionEvent): void
~handleOk(event: ActionEvent): void
-controllo(): boolean
~initialize(): void
operationpagecontroller
-u: Utente
-create: Button
-vote: Button
-Btngoback: Button
-btnResults: Button
-utenteDAO: IDAOUtenti
~requestResults(event: ActionEvent): void
~gobacck(event: ActionEvent): void
~handlecreate(event: ActionEvent): void
~handlevote(event: ActionEvent): void
+initialize(): void
pagecontroller
-login: Button
-registration: Button
~handlelogin(event: ActionEvent): void
~handleregistration(event: ActionEvent): void
resultsCategoricFormController
-u: Utente
-s: SessioneDiVoto
-sessionId: int = 0
-lblOutput: Label
-lblLogged: Label
-Btngoback: Button
-lblOutpudReferendum: Label
-sessioneDAO: SessioneIDAO
-utenteDAO: IDAOUtenti
-VotoDAO: IDAOVoto
~goback(event: ActionEvent): void
+initialize(): void
resultsOrdinaryFormController
-u: Utente
-s: SessioneDiVoto
-sessionId: int = 0
-bottomLabel: Label
-Btngoback: Button
-lblOutput: Label
-lblLogged: Label
-tableCandidates: TableView
-tableColumn: TableColumn
-sessioneDAO: SessioneIDAO
-utenteDAO: IDAOUtenti
-VotoDAO: IDAOVoto
~goback(event: ActionEvent): void
+updateColumns(s: SessioneDiVoto): void
+initialize(): void

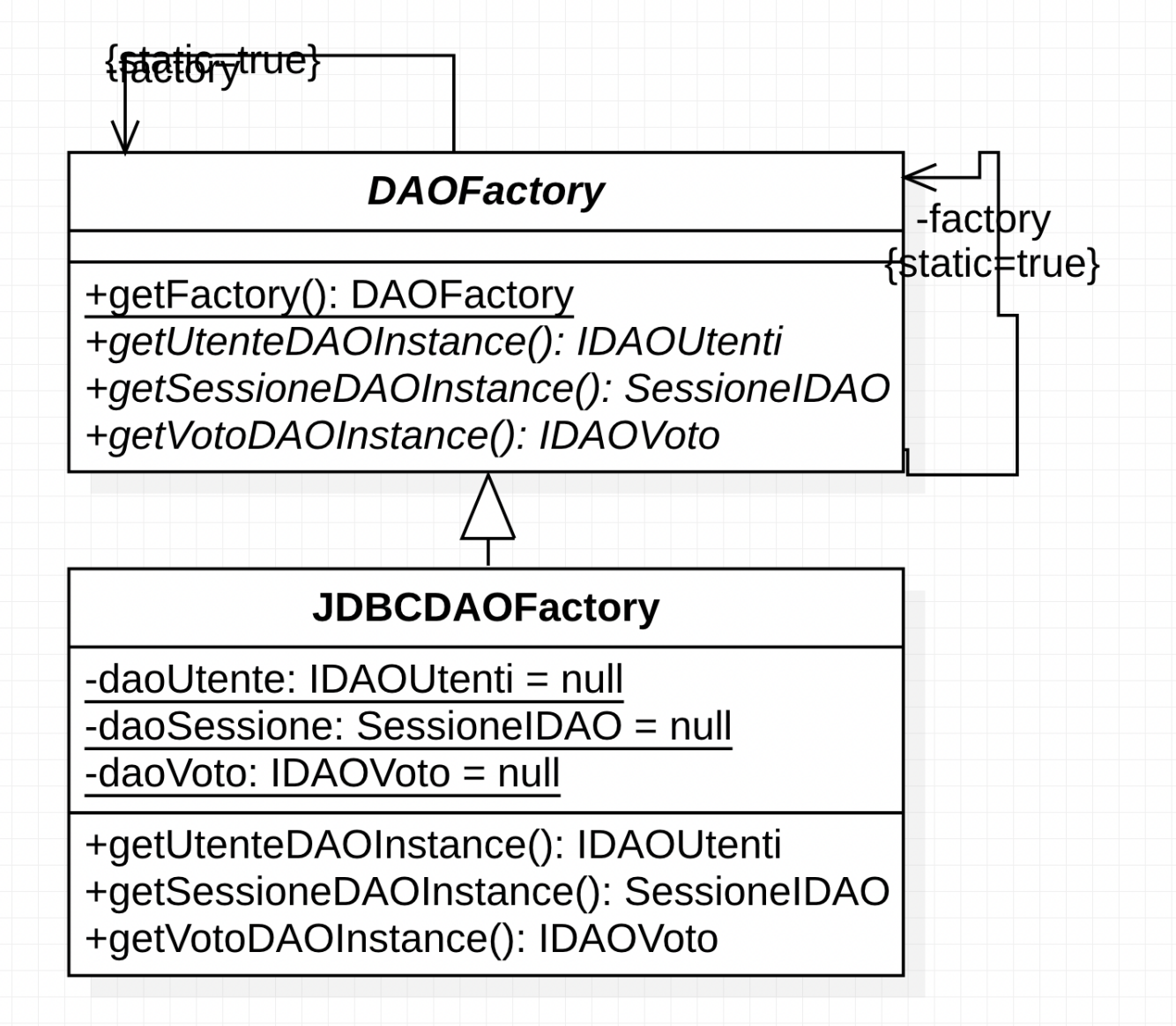
4.0.2 DAOs



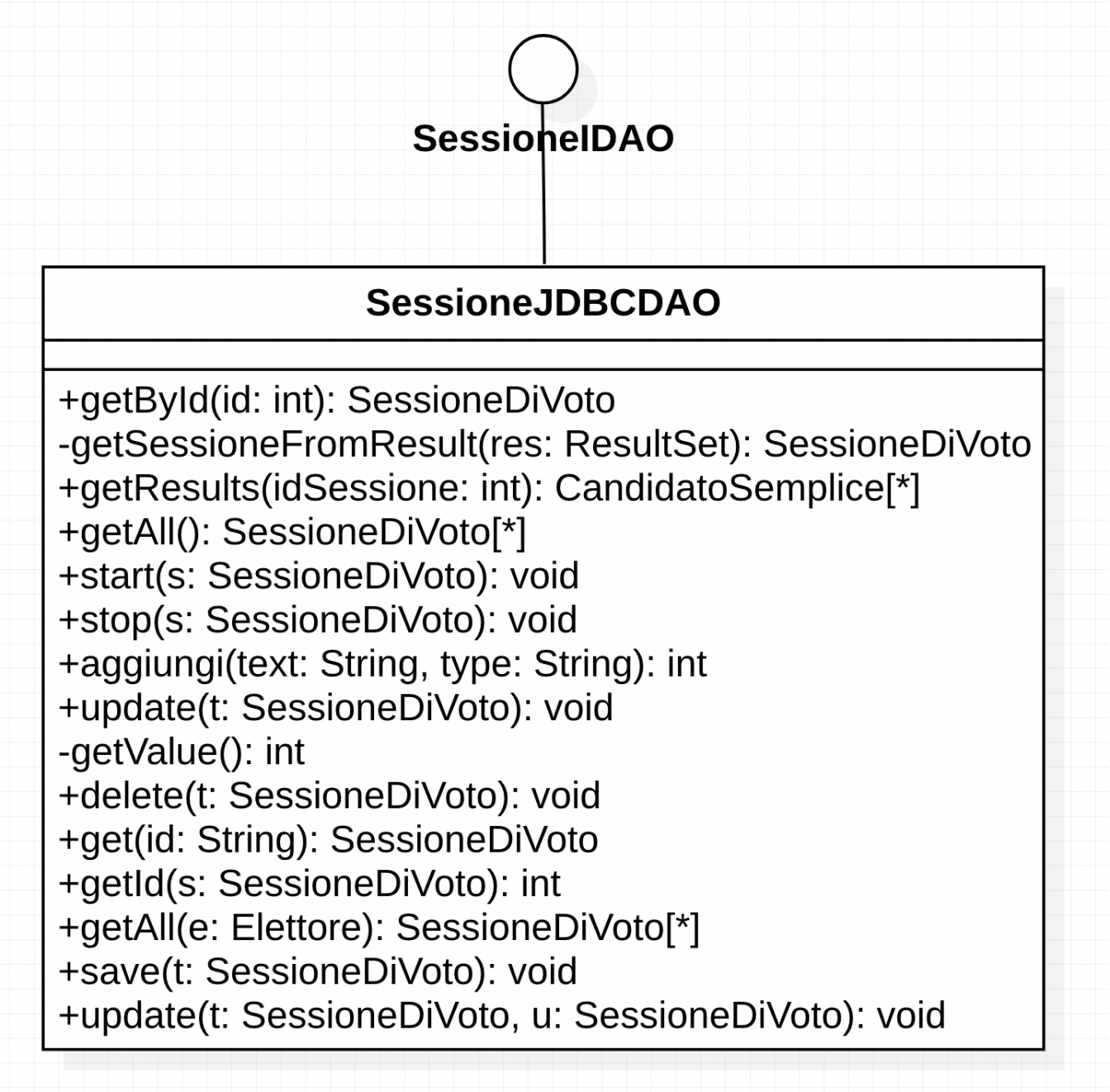
4.0.3 DAOs relationship



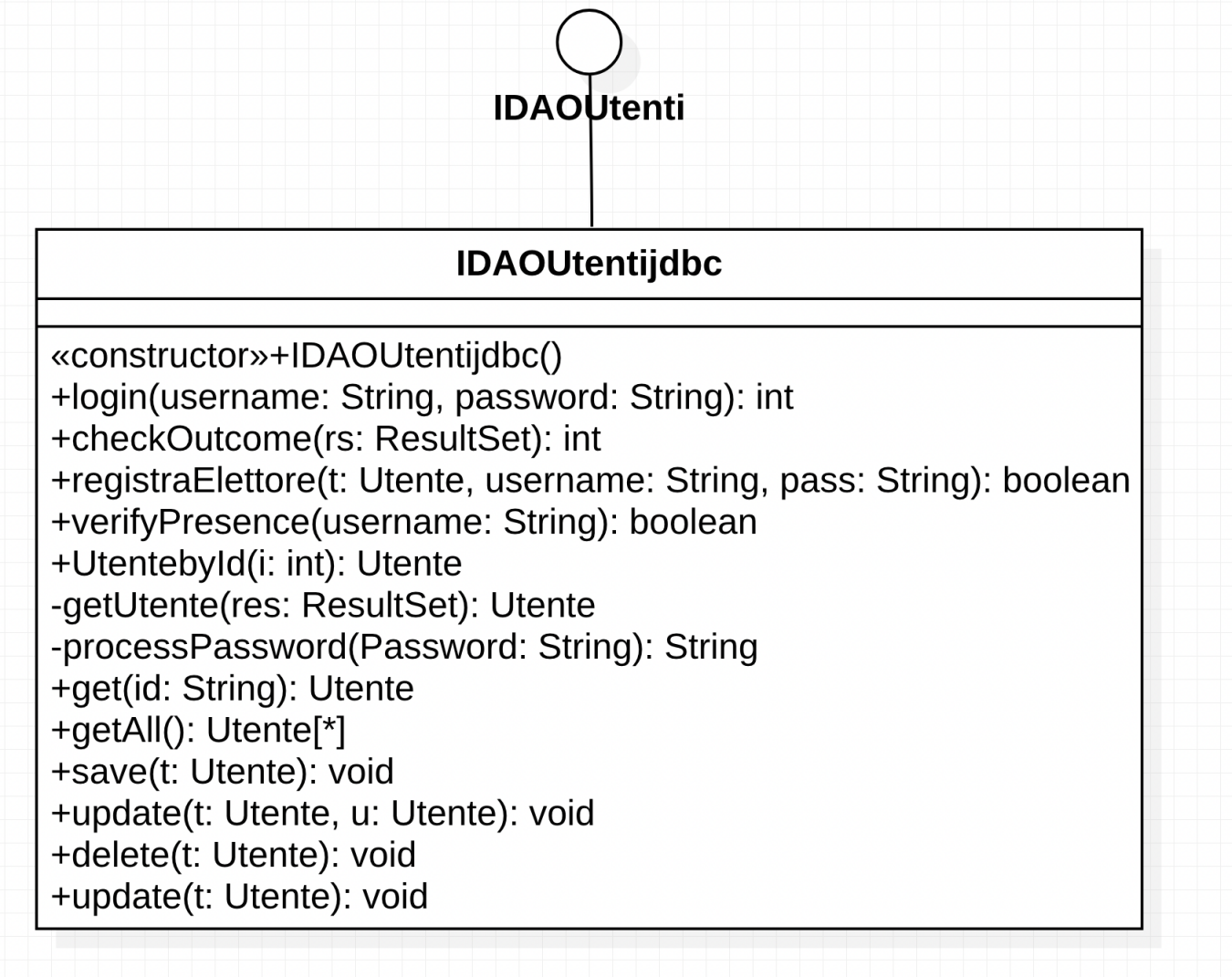
4.0.4 DAO Factory



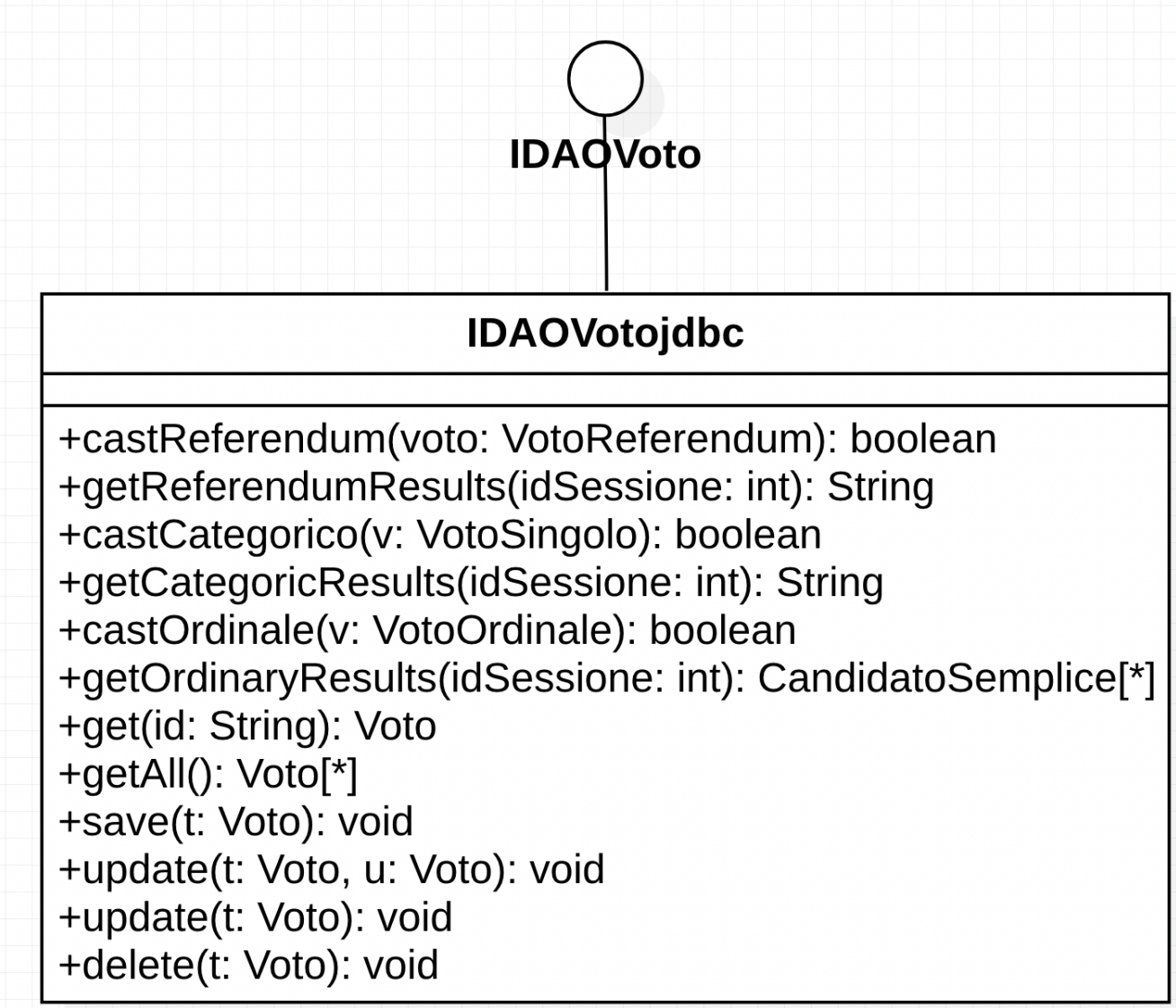
4.0.5 Sessione JDBCDAO



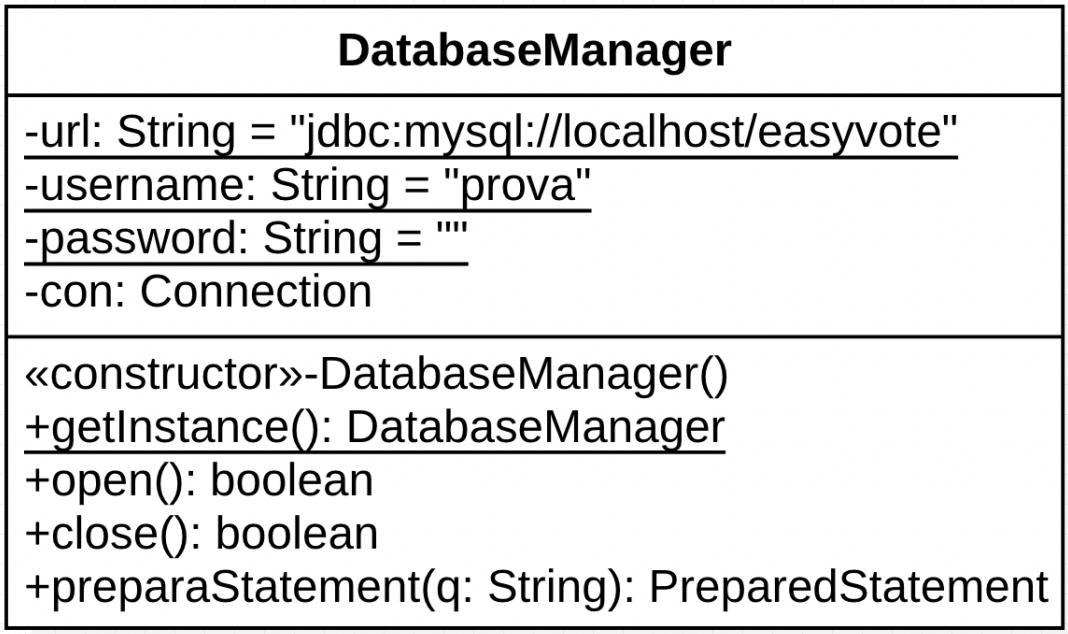
4.0.6 IDAOUtenti



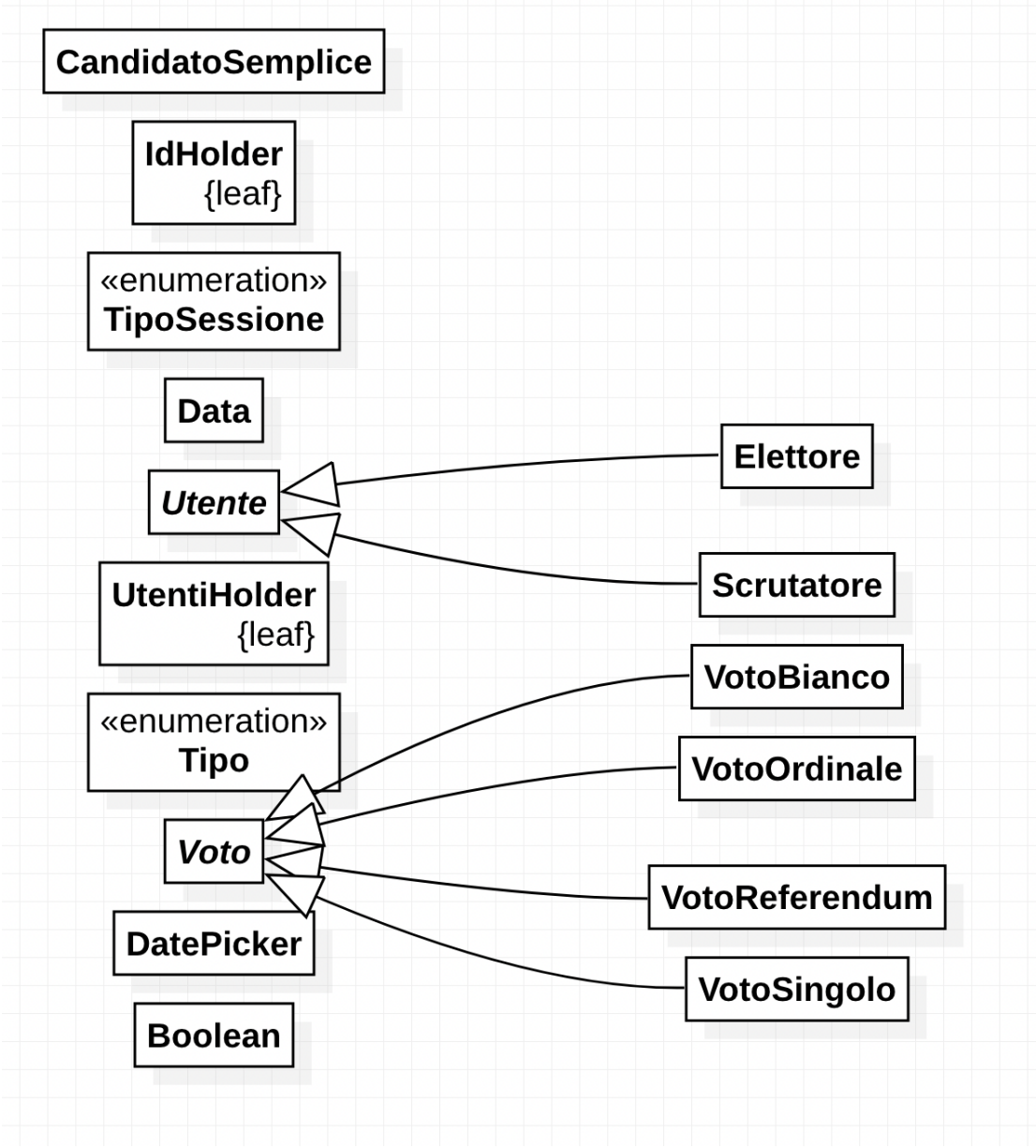
4.0.7 IDAOVoto



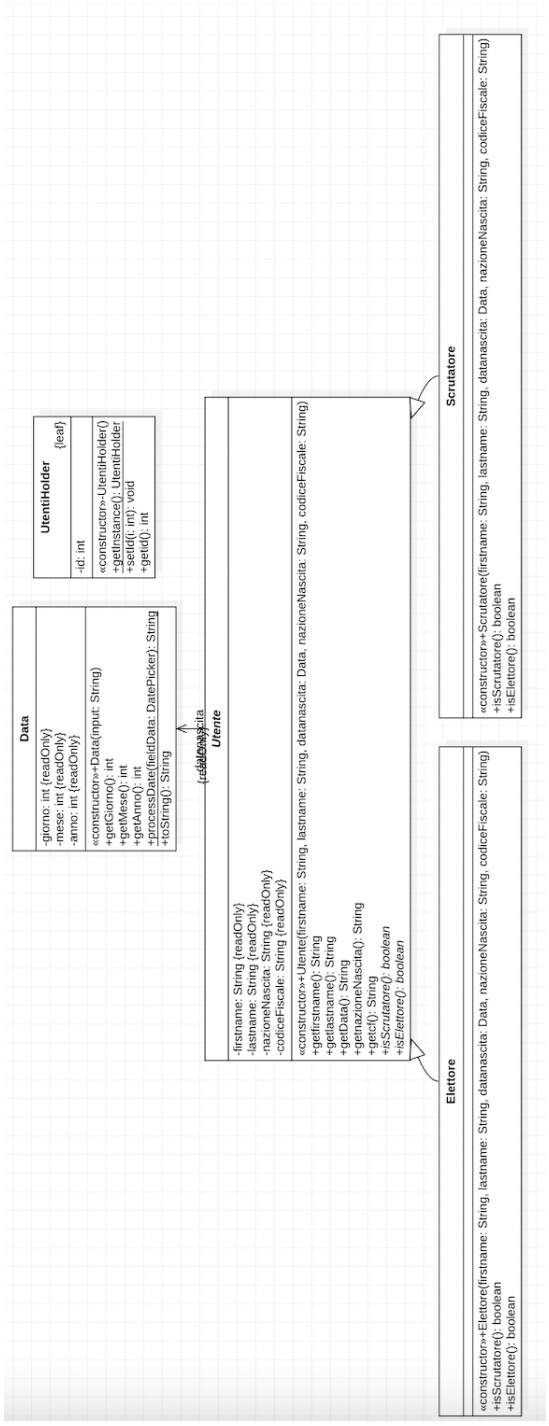
4.0.8 Database Manager



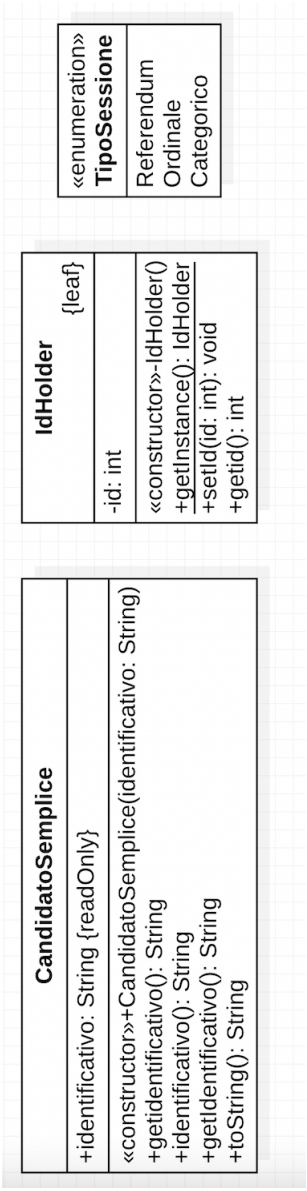
4.0.9 Utenti & Voto



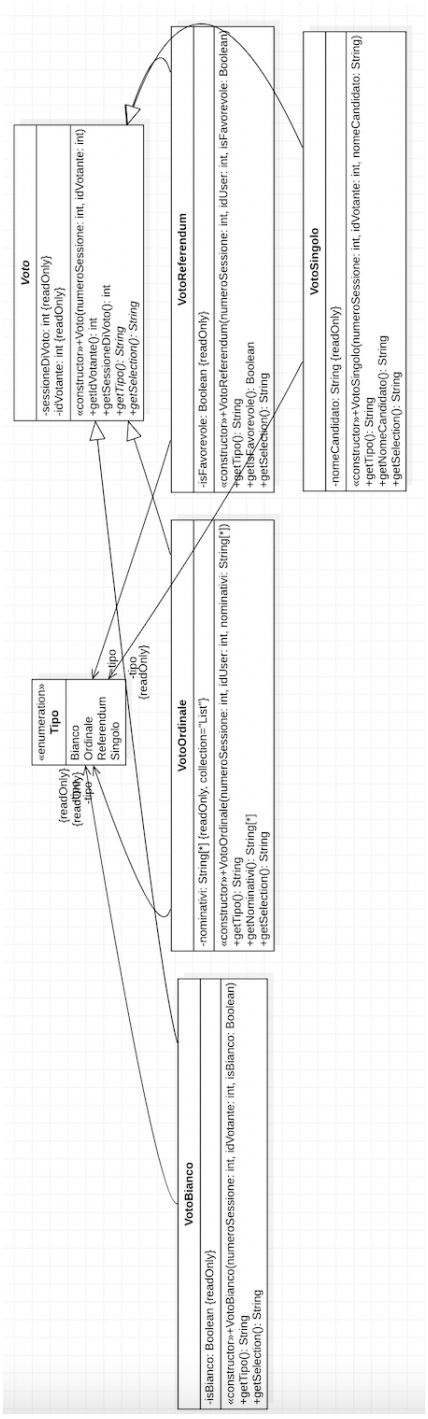
4.0.10 Utente


Data
-giorno: int {readOnly}
-mese: int {readOnly}
-anno: int {readOnly}
«constructor»+Data(input: String)
+getGiorno(): int
+getMese(): int
+getAnno(): int
+processDate(fieldData: DatePicker): String
+toString(): String
UtentiHolder
{leaf}
-id: int
«constructor»-UtentiHolder()
+getInstance(): UtentiHolder
+setId(i: int): void
+getId(): int
{ordered}
Utente
-firstname: String {readOnly}
-lastname: String {readOnly}
-nazioneNascita: String {readOnly}
-codiceFiscale: String {readOnly}
«constructor»+Utente(firstname: String, lastname: String, datanascita: Data, nazioneNascita: String, codiceFiscale: String)
+getFirstname(): String
+getLastname(): String
+getData(): String
+getNazioneNascita(): String
+getCf(): String
+isScrutatore(): boolean
+isElettore(): boolean
Elettore
«constructor»+Elettore(firstname: String, lastname: String, datanascita: Data, nazioneNascita: String, codiceFiscale: String)
+isScrutatore(): boolean
+isElettore(): boolean
Scrutatore
«constructor»+Scrutatore(firstname: String, lastname: String, datanascita: Data, nazioneNascita: String, codiceFiscale: String)
+isScrutatore(): boolean
+isElettore(): boolean

4.0.11 Session Holder

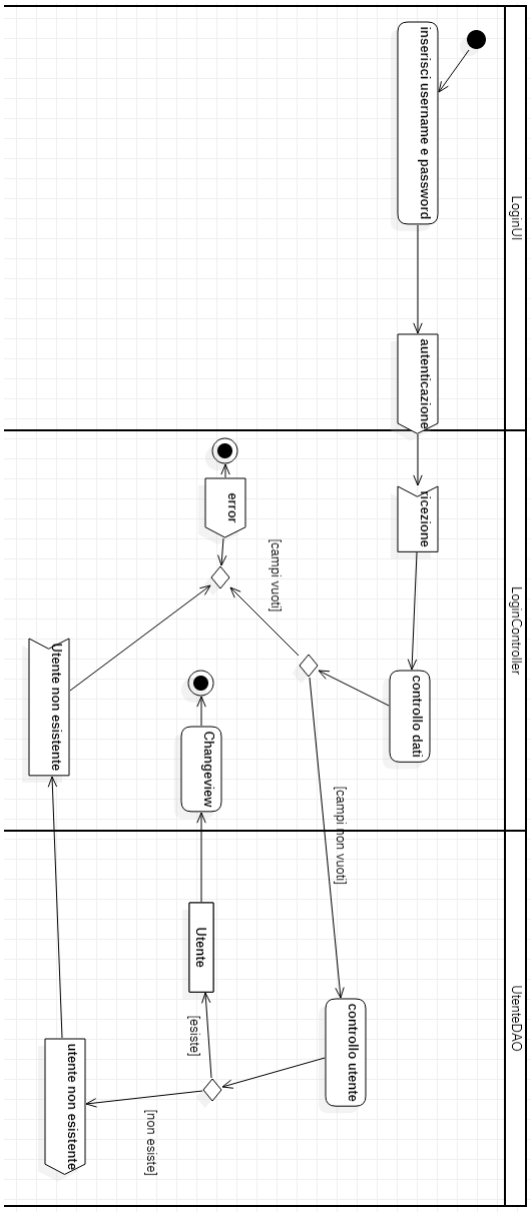


4.0.12 Voto


Voto
-sessioneDiVoto: int {readOnly}
-idVotante: int {readOnly}
«constructor»+Voto(numeroSessione: int, idVotante: int)
+getIdVotante(): int
+getSessioneDiVoto(): int
+getTipo(): String
+getSelection(): String
«enumeration»
Tipo
Bianco
Ordinale
Referendum
Singolo
tipo
-tipo
{readOnly}
{readOnly}
{readOnly}
-tipo
VotoReferendum
-isFavorevole: Boolean {readOnly}
«constructor»+VotoReferendum(numeroSessione: int, idUser: int, isFavorevole: Boolean)
+getTipo(): String
+getIsFavorevole(): Boolean
+getSelection(): String
VotoOrdinale
-nominativi: String[*] {readOnly, collection="List"}
«constructor»+VotoOrdinale(numeroSessione: int, idUser: int, nominativi: String[*])
+getTipo(): String
+getNominativi(): String[*]
+getSelection(): String
VotoSingolo
-nomeCandidato: String {readOnly}
«constructor»+VotoSingolo(numeroSessione: int, idVotante: int, nomeCandidato: String)
+getTipo(): String
+getNomeCandidato(): String
+getSelection(): String
VotoBianco
-isBianco: Boolean {readOnly}
«constructor»+VotoBianco(numeroSessione: int, idVotante: int, isBianco: Boolean)
+getTipo(): String
+getSelection(): String

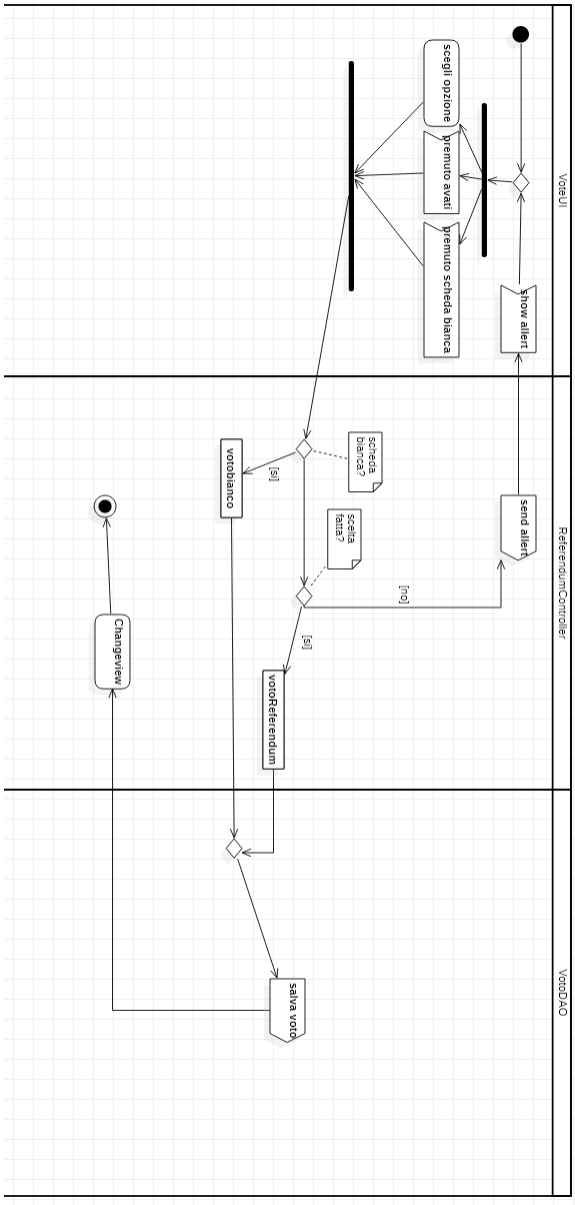
5 Sequence diagrams

5.1 Login

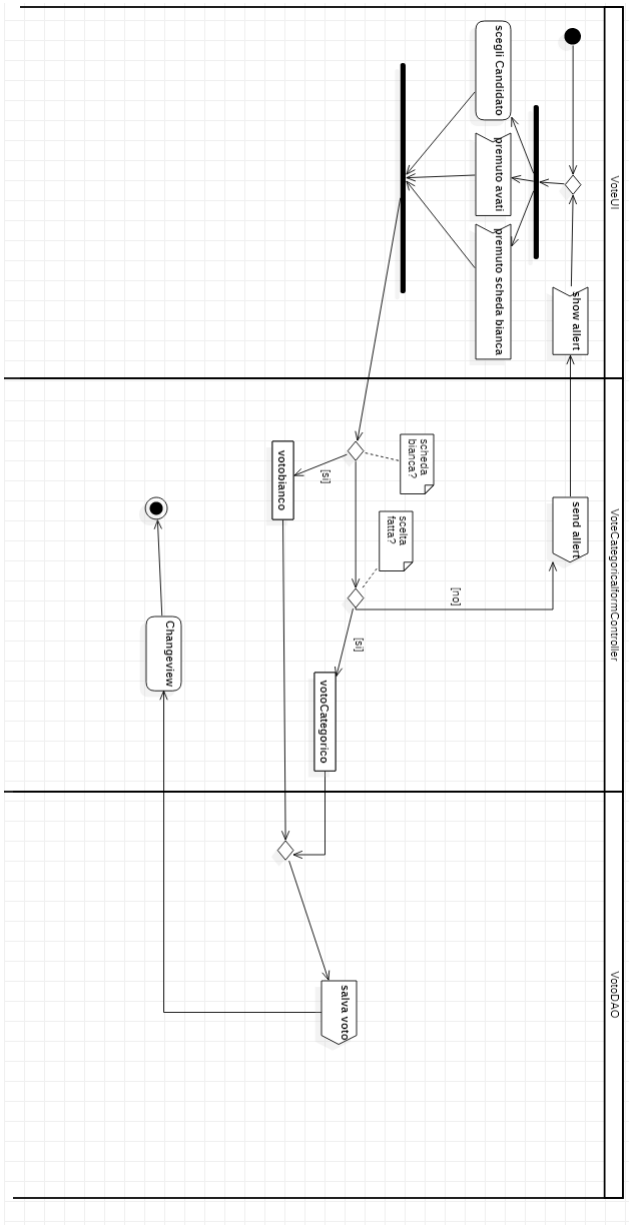


5.2 Vote

5.2.1 Referendum

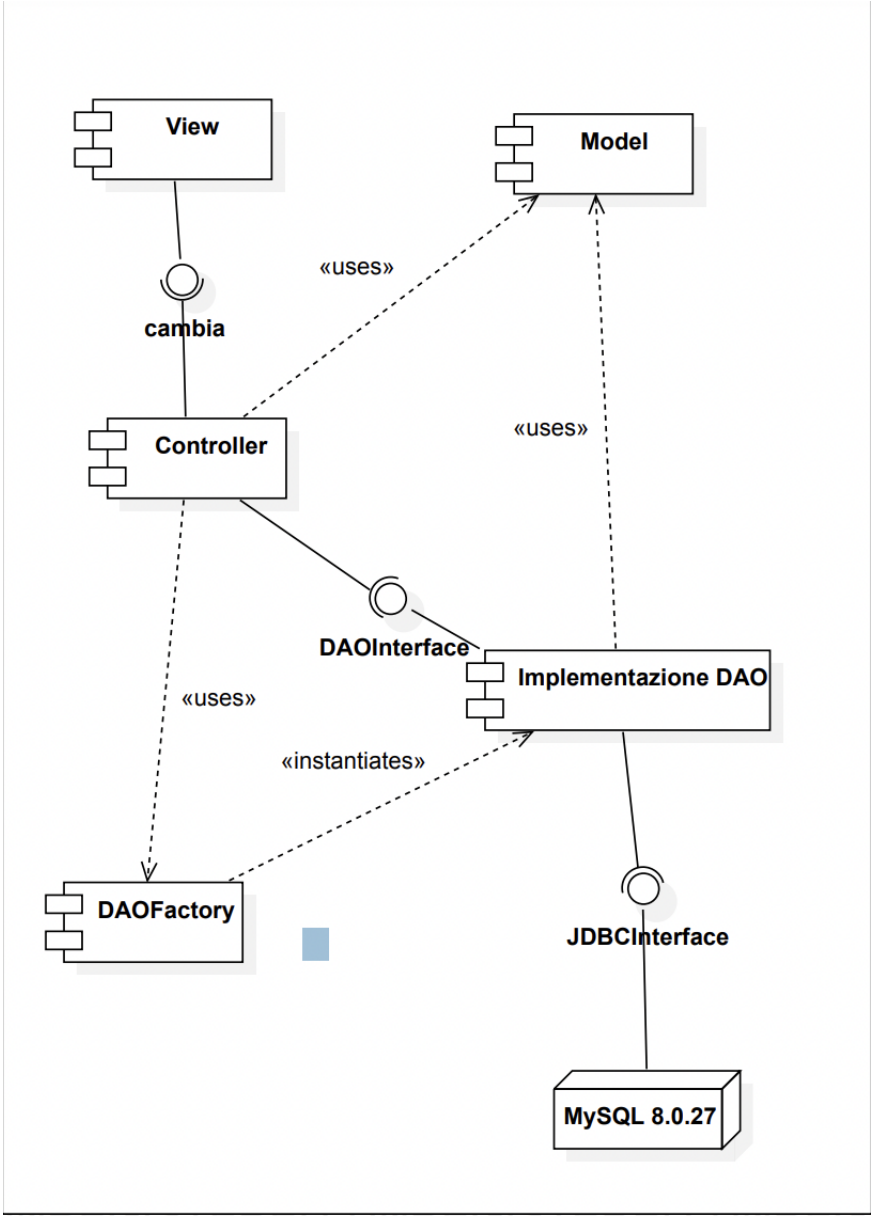


5.2.2 Categorical



6 Components diagrams

6.1 An overview of the components and their interactions



7 Design patterns employed

7.1 Singleton

The "Singleton" pattern is a software design pattern which aims to restrict the istantiation of a class to a single instance. One object is instantiated and spread across classes to coordinate actions.
The singleton pattern is utilized to spread the *SessioneDiVoto* with the *SessioneIDAO* class and the *Utente* with the *IDAOUtenti* class.

7.2 DAO

The "DAO pattern" o "Data Access Object" is structural pattern to isolate the application layer from the database layer.

The "DAO pattern" is combined with the Abstract factory pattern to hide complexities related to accessing databases to specific classes, leaving there the definition of specific implementations. The "DAO pattern" are organized under *easyVote.DAO*, and are *easyVote.DAO*, *easyVote.DAO.factory*, *easyVote.DAO.sessione*, *easyVote.DAO.utenti*, *easyVote.DAO.voto*, *easyVote.database*.

7.3 MVC

The "MVC model" o "Model-View-Controller" is a software design pattern used to develop user interfaces that divides program logic into three interconnected elements.

The easyVote software is divided in models, controllers and views. The file structure is organized under *easyVote.models*, *easyVote.controllers*, *easyVote.views*, *easyVote.DAO*, *easyVote.database* and *easyVote.factory*. This is done to separate internal representations and to present data in intuitive and easy to use way to both the programmer and the user.

7.4 Abstract factory

The "Abstract factory" is a design pattern which aims to encapsulate object creation in a separate object, this is done to facilitate the implementation of new features. The easyVote software used this design pattern to facilitate the creation of new "DAO" classes.

8 Database description

The easyVote database was designed from the ground up with simplicity and easy of use in mind. The DBMS chosen was the "MySQL" Database due to its widespread use, ease of connectivity, and the way the "JDBC" library allows for a seamless experience.

The tables present in the easyVote Database are: "voto", "session", "users".

8.1 users

The "users" table's job is to encapsulate all the information regarding a easyVote user such as:

- iduser: a user's automatically generated id number;
- name: a user's inserted name;
- lastname: a user's inserted surname;
- birthdate: a user's inserted birthdate;
- codicefiscale: a user's inserted and system checked "codice fiscale" or vat code number;
- username: a user's chosen username used to login;
- password: a user's chosen password hashed using the "SHA-256" secure hashing algorithm;
- isadmin: to identify if the user is an admin or not.

The primary key is "iduser" and there are no secondary keys.

The database state at the moment of testing was:

iduser	name	lastname	birthdate	birthplace	codicefiscale	username	password	isadmin
19	federico	zhou	2022-03-17	itayl	zhsjandmsjx...	blabla	blabla	0
20	fsdafasd	ffasd	2022-03-10	fasd	FDSASDFGH...	fsdafas	600bd0c...	0
21	Alessia	Pena	2022-03-16	america	AMSKDLFM...	ale	7ba0ea6...	0
24	federico	blabla	2022-05-02	ita	123456789...	bloblo	314bd86...	1
25	ffdsdas	fsdafdsa	2022-06-08	fadsfads	123456789...	bleble	708b1a7...	0
26	fdsafads	fsadfdas	2022-06-01	fsdafdsa	123456789...	admin	8c6976e...	0

The "users" table

8.2 session

The "session" table's job is to encapsulate all the information regarding a vote session such as:

- id: a session's automatically generated id number;
- text: an admin inserted session text description;
- candidato: an admin inserted and system factored JSON type text which identifies the candidates;
- isOpen: a session's attribute to define if the session is open or not;
- type: an admin inserted attribute to identify the session's type.

The primary key is "id" and there are no secondary keys.

The database state at the moment of testing was:

id	text	candidati	isOpen	type
78	EMPTY	{"candidato1": "fede", "candidato2": "angelo", "candidato...	1	Ordinale
79	National Elections	{"candidato1": "Fede", "candidato2": "Angelo", "candidato...	1	Categorico
80	Elezione nazionale	{"candidato1": "Carlo", "candidato2": "Alberto", "candidat...	1	Ordinale
81	Elezioni europee	{"candidato1": "Angelo", "candidato2": "Micheal"}	1	Ordinale
82	Giuridice	NULL	1	Referendum

The "session" table

8.3 voto

The "voto" table's job is to encapsulate all the information regarding a session vote such as:

- idSession: a system inserted valid id session number;
- idUser: a system inserted valid idUser number;
- selection: a system generated JSON type text which identifies an idUser's selection for a idSession vote session.

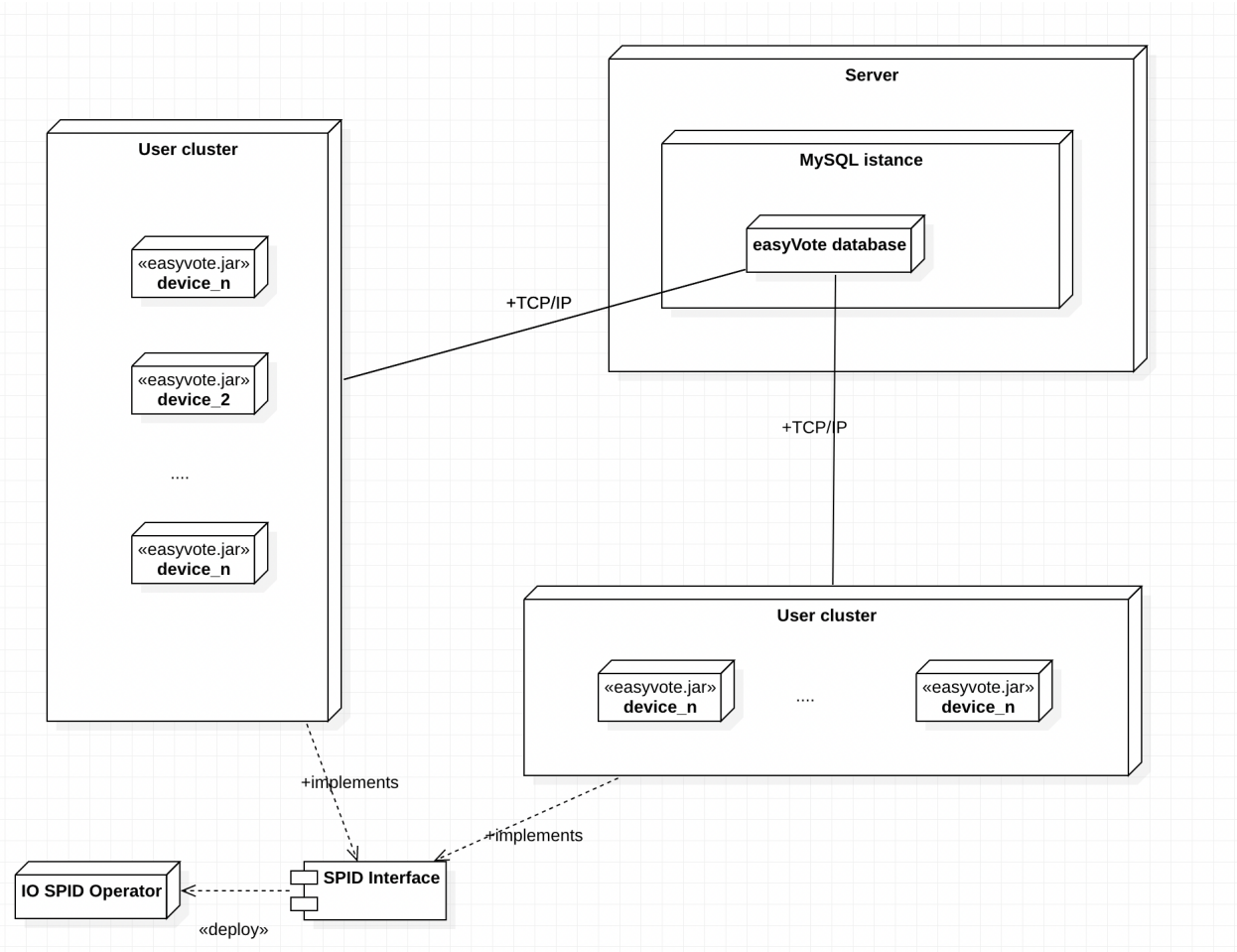
The primary keys are "idSession" and "isUser" and there are no secondary keys.

The database state at the moment of testing was:

idSession	idUser	selection
62	1	{"selection": "angelofedeaaa; fede"}
61	22	{"selection": "federico"}
62	22	{"selection": "1"}
66	24	{"selection": "fede; angelo; amelia; alessia"}
69	24	{"selection": "angelo"}
76	24	{"selection": "0"}
66	25	{"selection": "fede; amelia; angelo; alessia"}
69	25	{"selection": "federico"}
76	25	{"selection": "1"}
69	26	{"selection": "federico"}
76	26	{"selection": "1"}

The "voto" table

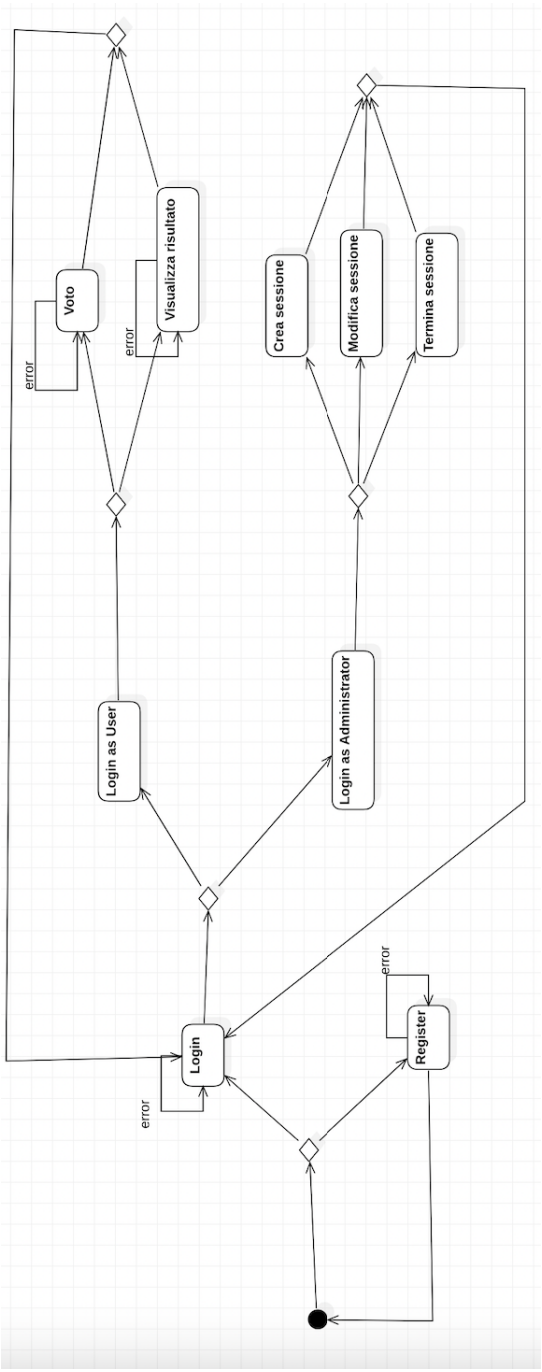
9 Deployment diagram



easyVote's deployment diagram

The SPID interface's job is to facilitate communication between easyVote applications and the "IO SPID Operator" which provides high level security authentication to easyVote's terminal.

10 UI description diagram



easyVote's UI description diagram

11 Specification and validation of JML

The signatures of functions and methods of the easyVote project were complemented with specification and validation using the JML and Javadoc (using the google style of Javadoc) a few snippets are:

- In the context of dates during the registration process:
  Invariants: self.giorno $>$ 0 and self.giorno $<$ 32, self.mese $>$ 0 and self.mese $<$ 13, self.anno $>$ 1800

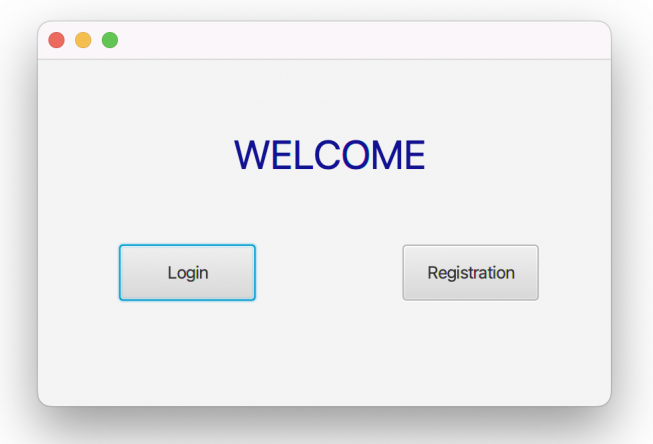
- In the context of dates during in the session definition process:
  Invariants: self.giorno $>$ 0 and self.giorno $<$ 32, self.mese == 2 implies self.giorno $<$ 30, self.mese == 4 or self.mese == 6 or self.mese == 9 or self.mese == 1

- in the context of users:
  Pre-conditions: username != null and password != null

12 User interface description

12.1 Landing page

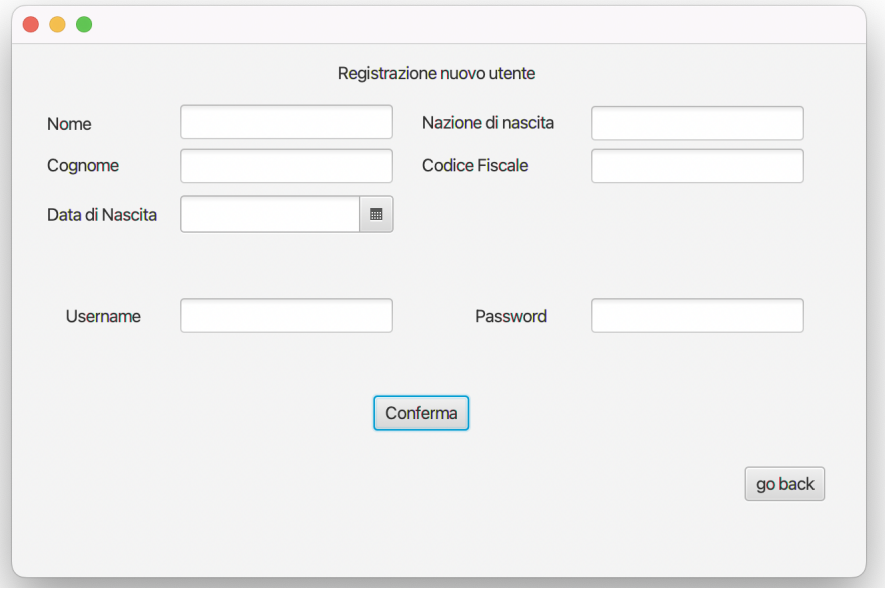
Form at the opening of the program



Landing page

12.2 Registration page

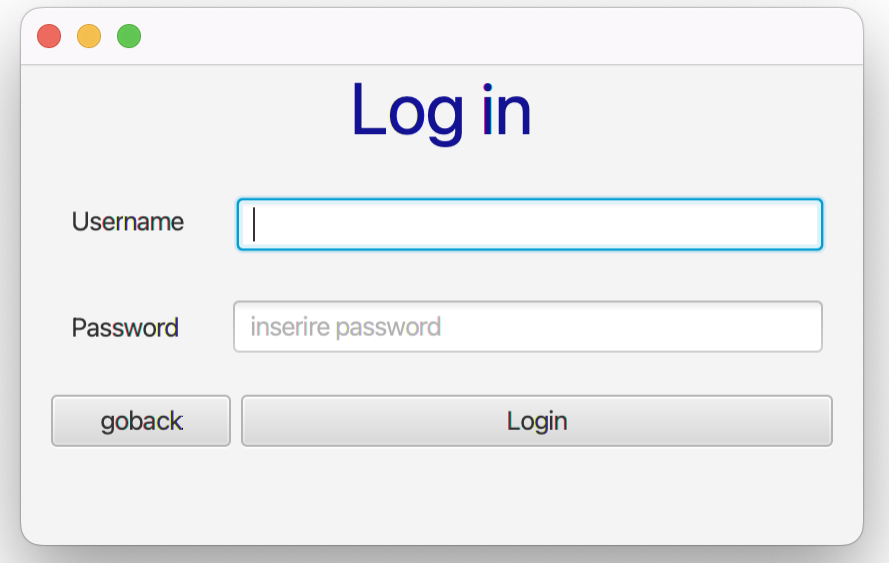
Form to register a new user



Registration page

12.3 Login page

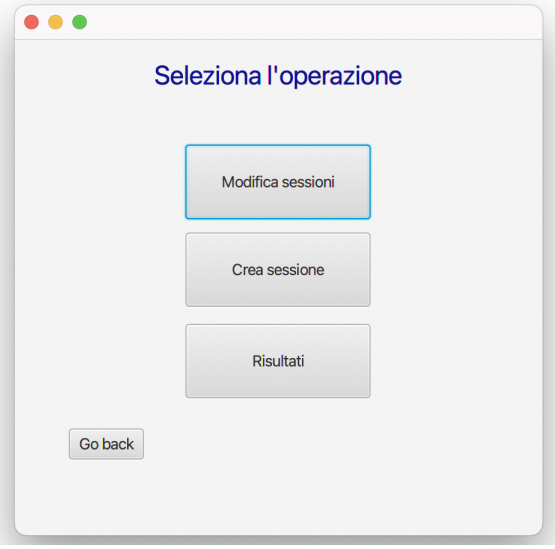
Form to login a user



Login page

12.4 Activity page

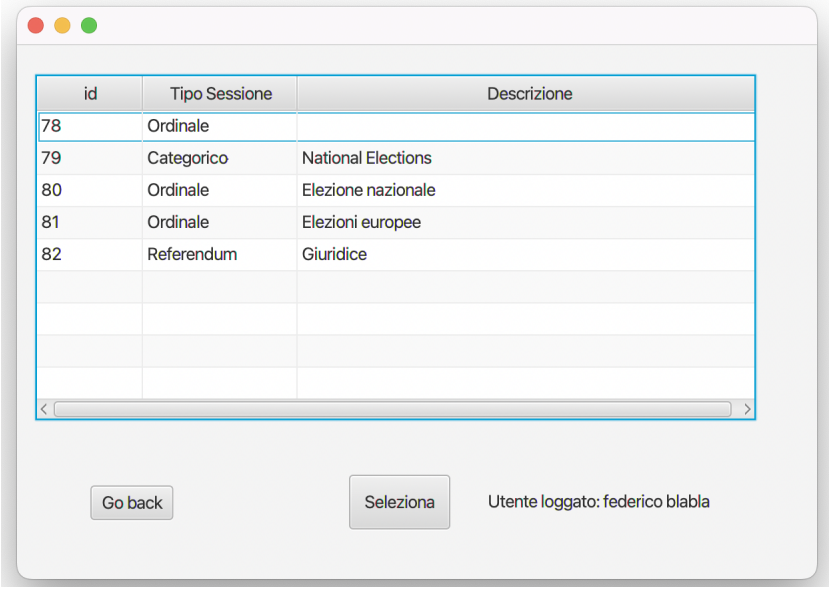
Form to select the activity that a user wants to make as an admin



Activity page

12.5 Session modification page

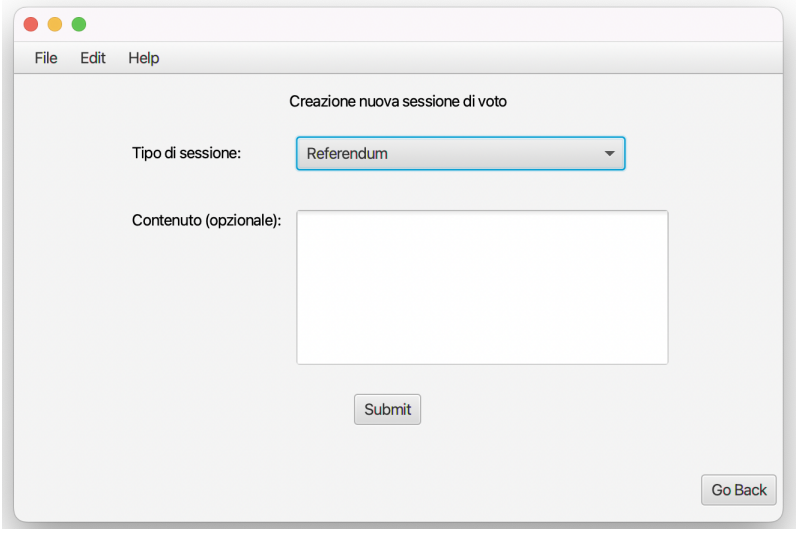
Form to select the session upon which an admin wants to make modifications



Session modification page

12.6 Session creation page

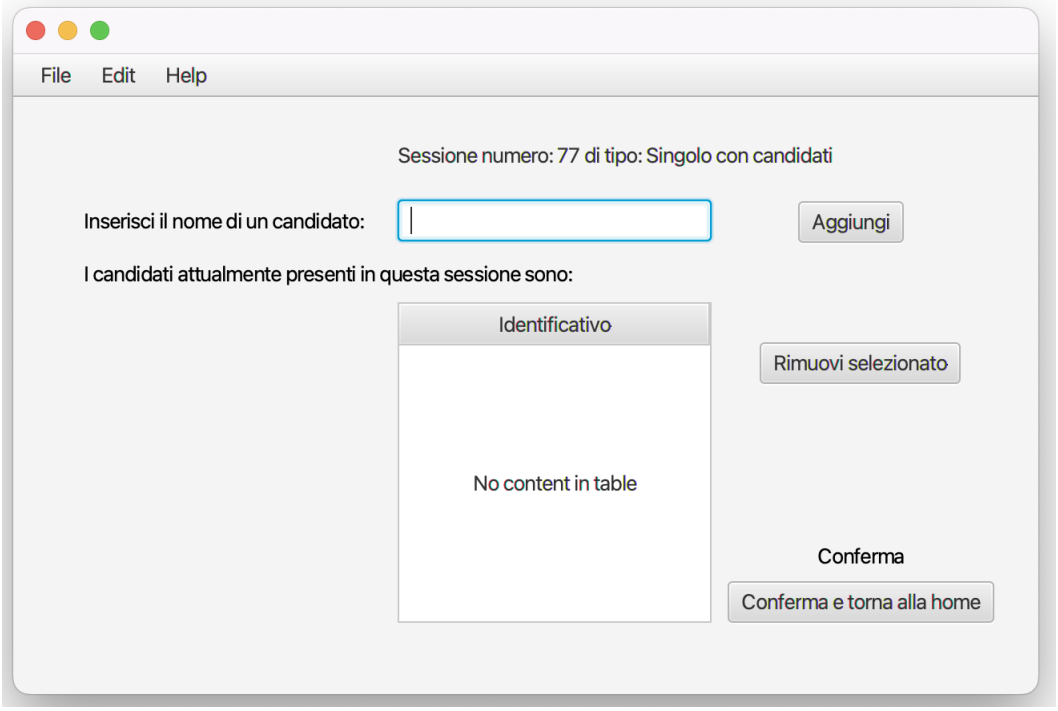
Form to create a new voting session



Session creation page

12.7 Session properties definition page

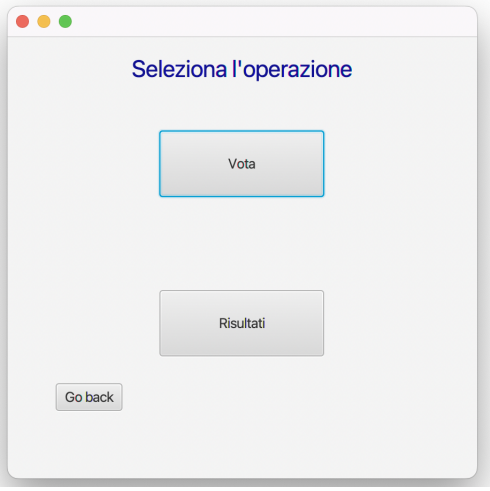
Form to set the candidates, description and remove candidates from a created voting session



Session properties definition page

12.8 Action selection page

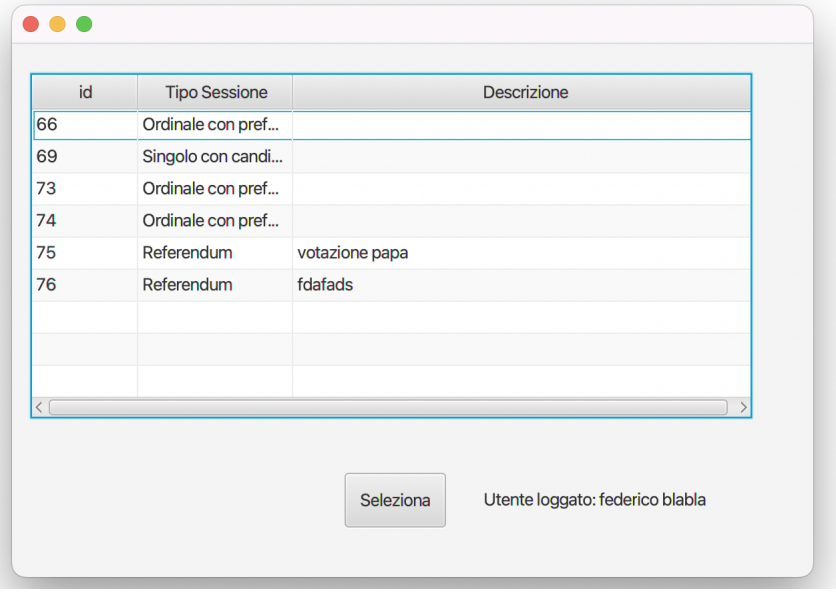
Form to select which operation a user wishes to make as logged user



Action selection page

12.9 Session selection page

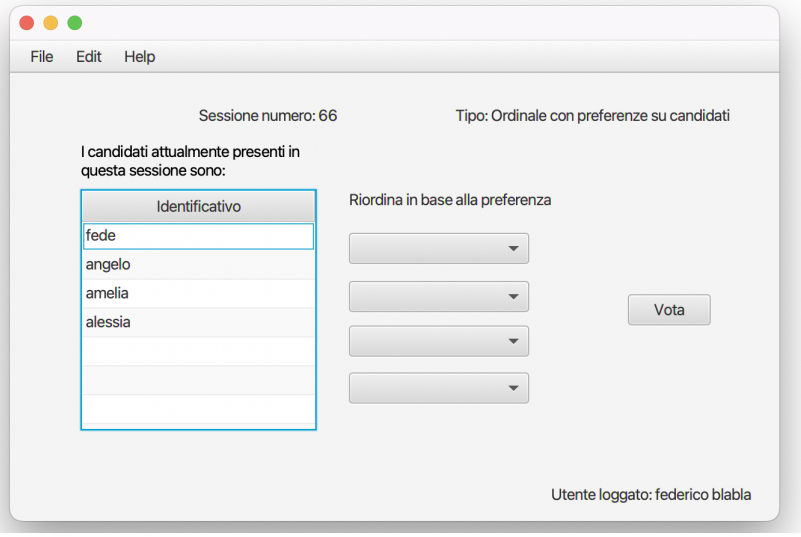
Form to select the session in which the logged user wants to vote in



Session selection page

12.10 Preferential vote page

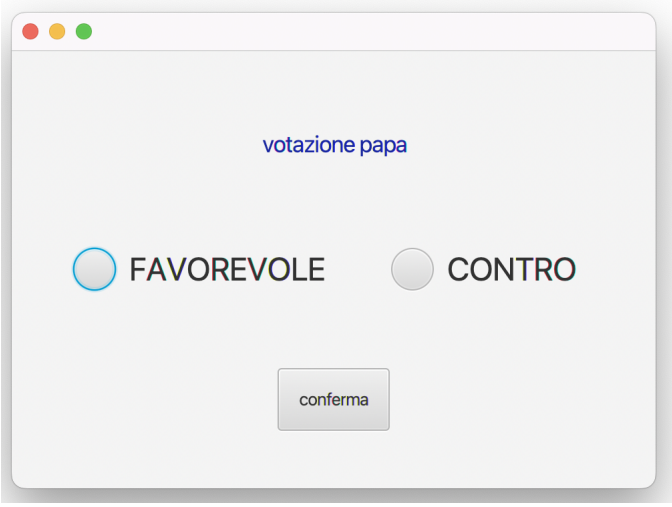
Form to vote in a preferential voting session as a logged user



Preferential vote page

12.11 Referendum vote page

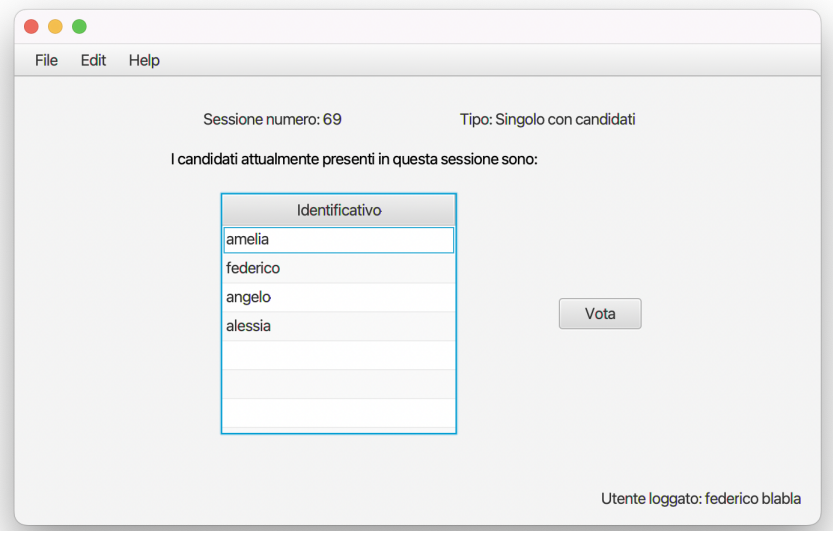
Form to vote in a referendum voting session as a logged user



Referendum vote page

12.12 Categorical vote page

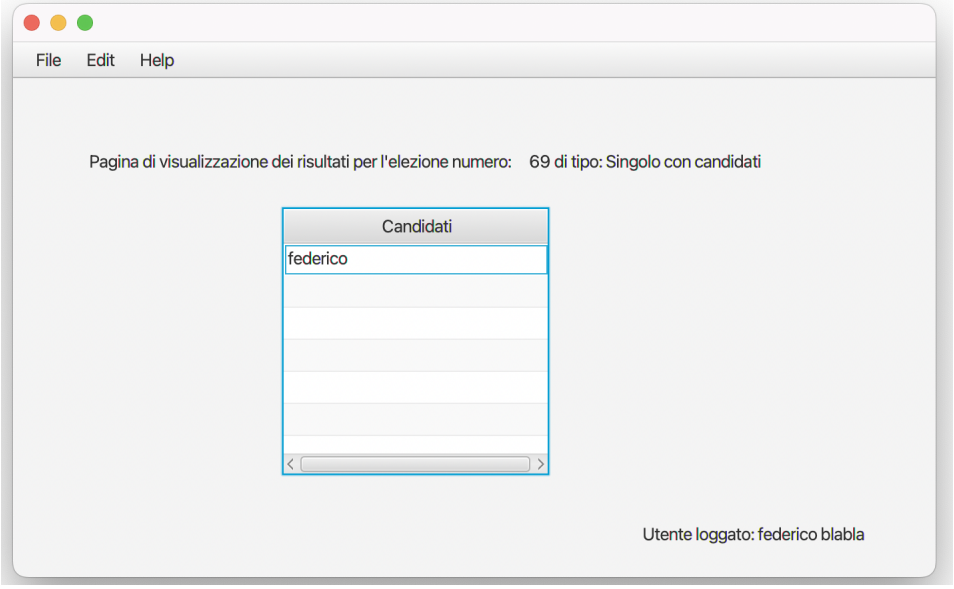
Form to vote in an categorical voting session as a logged user



Categorical vote page

12.13 Categorical vote results page

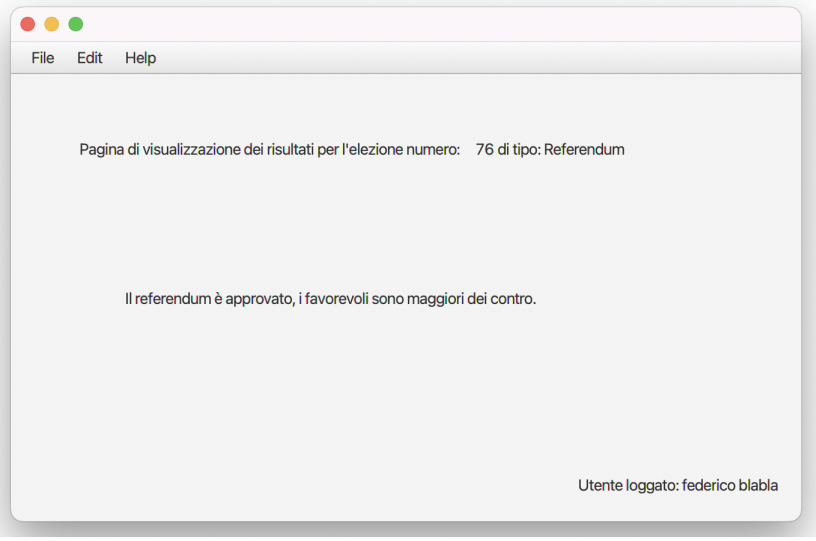
Form to display the results of an categorical voting session as an admin



Categorical vote results page

12.14 Referendum vote results page

Form to display the results of a referendum voting session as an admin



Referendum vote results page

13 Testing

The easyVote project was tested thoroughly and extensively with JUnit. Testing was done on the following aspects:

- User registration
- User login
- User deletion
- Admin registration
- Admin login
- Admin deletion
- Open session visualization
- Session creation
- Session modification
- Session deletion
- Voting in a referendum
- Voting in an ordinary session
- voting in a simple voting session
- Session results display

and many more.

14 Installation mode and settings

The easyVote project can be found in the Github repository. Please follow these tips:

- Make sure you are using Eclipse as your IDE.
- Make sure to build using the following "VM Arguments": *–module-path "yourPath" –add-modules javafx.controls,javafx.fxml*, where "yourPath" refers to your path to the JavaFX folder.
- If the program is unable to start, try to call the command: *mvn eclipse:eclipse*
- Import the "MySQL" database under "/easyVote/database". You can access administrator utilities using the username "bloblo" and password "bloblo".