Web technologies

Progetto "**MeetingsApp**"

RIA

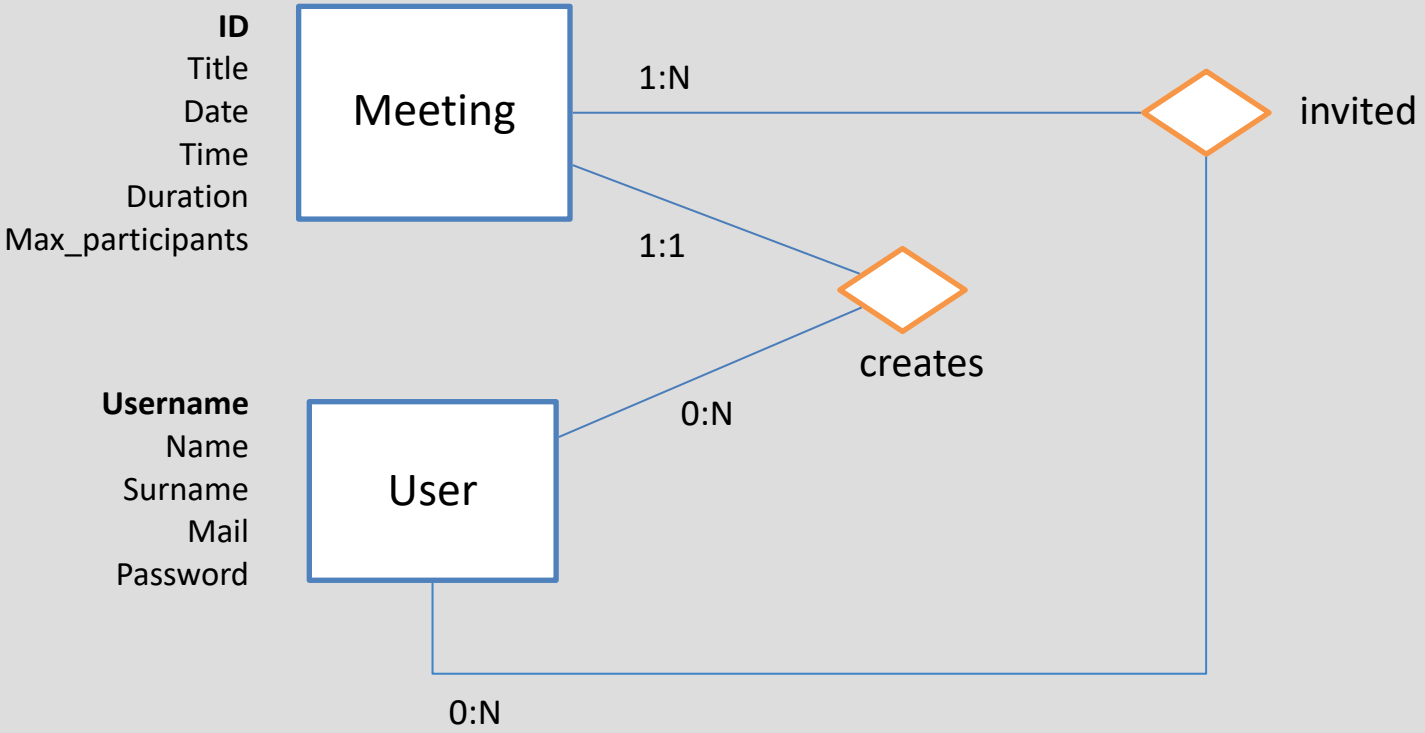
Luca Minotti

Matricola 886716
Codice Persona 1052339

Si realizzi un’applicazione client server web che modifica le specifiche come segue:

- L’applicazione supporta registrazione e login mediante una pagina pubblica con opportune form. La registrazione controlla la validità sintattica dell’indirizzo di email e l’uguaglianza tra i campi “password” e “ripeti password”, anche a lato client. La registrazione controlla l’unicità dello username.
- Dopo il login, l’intera applicazione è realizzata con un’unica pagina.
- Ogni interazione dell’utente è gestita senza ricaricare completamente la pagina, ma produce l’invocazione asincrona del server e l’eventuale modifica del contenuto da aggiornare a seguito dell’evento.
- La scelta dall’anagrafica deve essere realizzata con una pagina modale con i bottoni invia e cancella.
- I controlli di correttezza del numero di invitati e del massimo numero di tentativi, con i relativi messaggi di avvertimento, devono essere realizzati anche a lato client.
- Lo stato dell’interazione (numero di tentativi) deve essere memorizzato a lato client.

Database **design**



Local **database** schema

```
CREATE TABLE `user` (
  `username` VARCHAR(15) NOT NULL,
  `password` VARCHAR(32) NOT NULL,
  `name` VARCHAR(25) NOT NULL,
  `surname` VARCHAR(25) NOT NULL,
  `mail` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`username`));

 CREATE TABLE `meeting` (
   `id` INT NOT NULL AUTO_INCREMENT,
   `title` VARCHAR(45) NOT NULL,
   `date` DATE NOT NULL,
   `time` TIME NOT NULL,
   `duration` INT NOT NULL,
   `max_participants` INT NOT NULL,
   `creator` VARCHAR(15) NOT NULL,
   CONSTRAINT `username` FOREIGN KEY (`creator`)
   REFERENCES `user`(`username`)
   ON DELETE CASCADE ON UPDATE CASCADE,
   PRIMARY KEY (`id`));
```

Local **database** schema

```
CREATE TABLE `invitation` (
`id_meeting` INT NOT NULL,
`guest` VARCHAR(45) NOT NULL,
CONSTRAINT `id` FOREIGN KEY (`id_meeting`)
REFERENCES `meeting`(`id`)ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT `guest` FOREIGN KEY (`guest`)
REFERENCES `user`(`username`)ON DELETE CASCADE ON UPDATE CASCADE,
PRIMARY KEY (`id_meeting`,`username`));
```

Application **requirements** analysis

Si realizzi un'applicazione client server web che modifica le specifiche come segue:

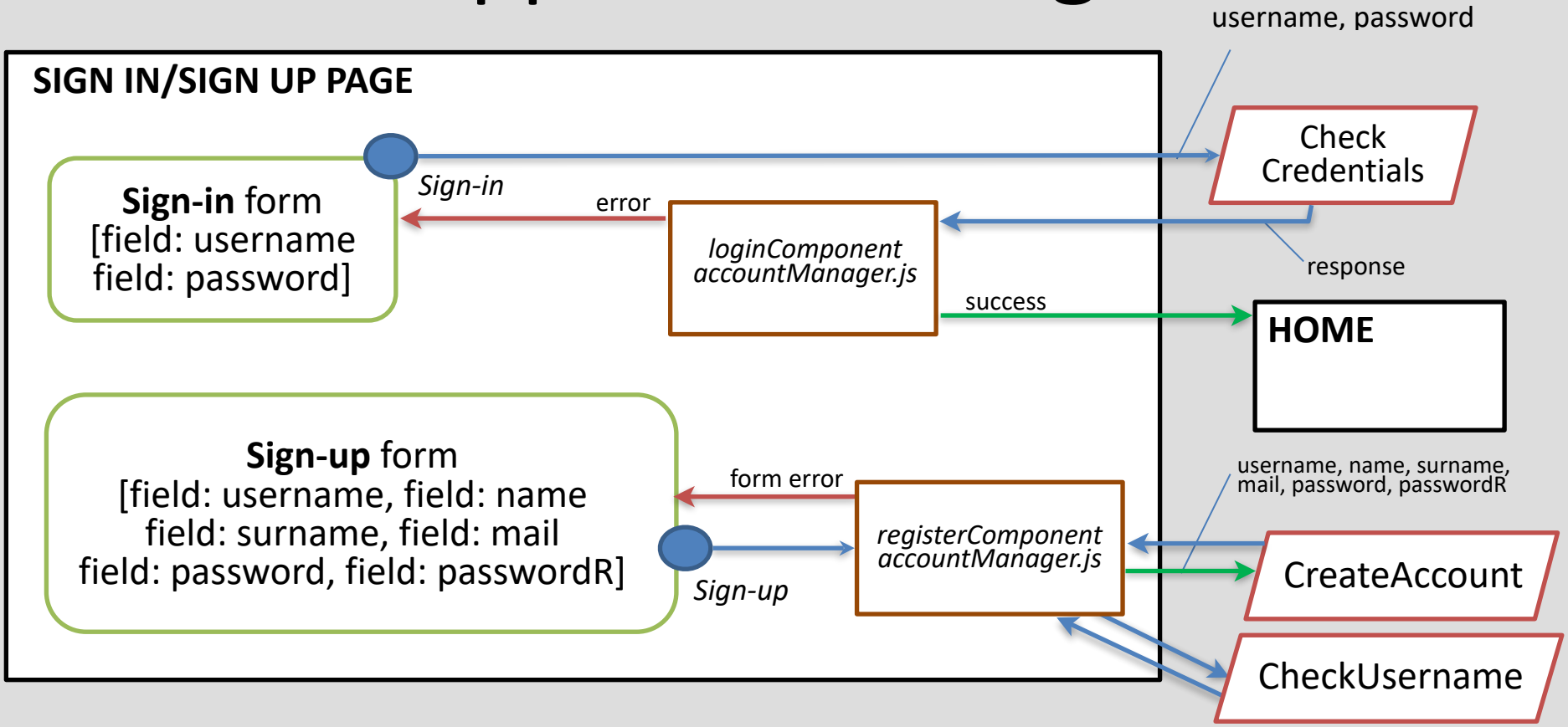
- L'applicazione supporta **registrazione** e **login** mediante una **pagina pubblica** con opportune **form**. La registrazione controlla la validità sintattica dell'indirizzo di email e l'uguaglianza tra i campi "password" e "ripeti password", anche a lato client. La registrazione controlla l'unicità dello username.
- Dopo il login, l'intera applicazione è realizzata con un'**unica pagina** [che mostra la **lista delle riunioni indette** dall'utente, la **lista delle riunioni a cui è stato invitato** e un **form** per la creazione di un nuovo meeting]
- Ogni interazione dell'utente è gestita senza ricaricare completamente la pagina, ma produce l'invocazione asincrona del server e l'eventuale modifica del contenuto da aggiornare a seguito dell'evento.
- La **scelta** dall'anagrafica deve essere realizzata con una pagina modale con i **bottoni invia** e **cancella**.
- I controlli di correttezza del numero di invitati e del massimo numero di tentativi, con i relativi messaggi di avvertimento, devono essere realizzati anche a lato client.
- Lo stato dell'interazione (numero di tentativi) deve essere memorizzato a lato client.

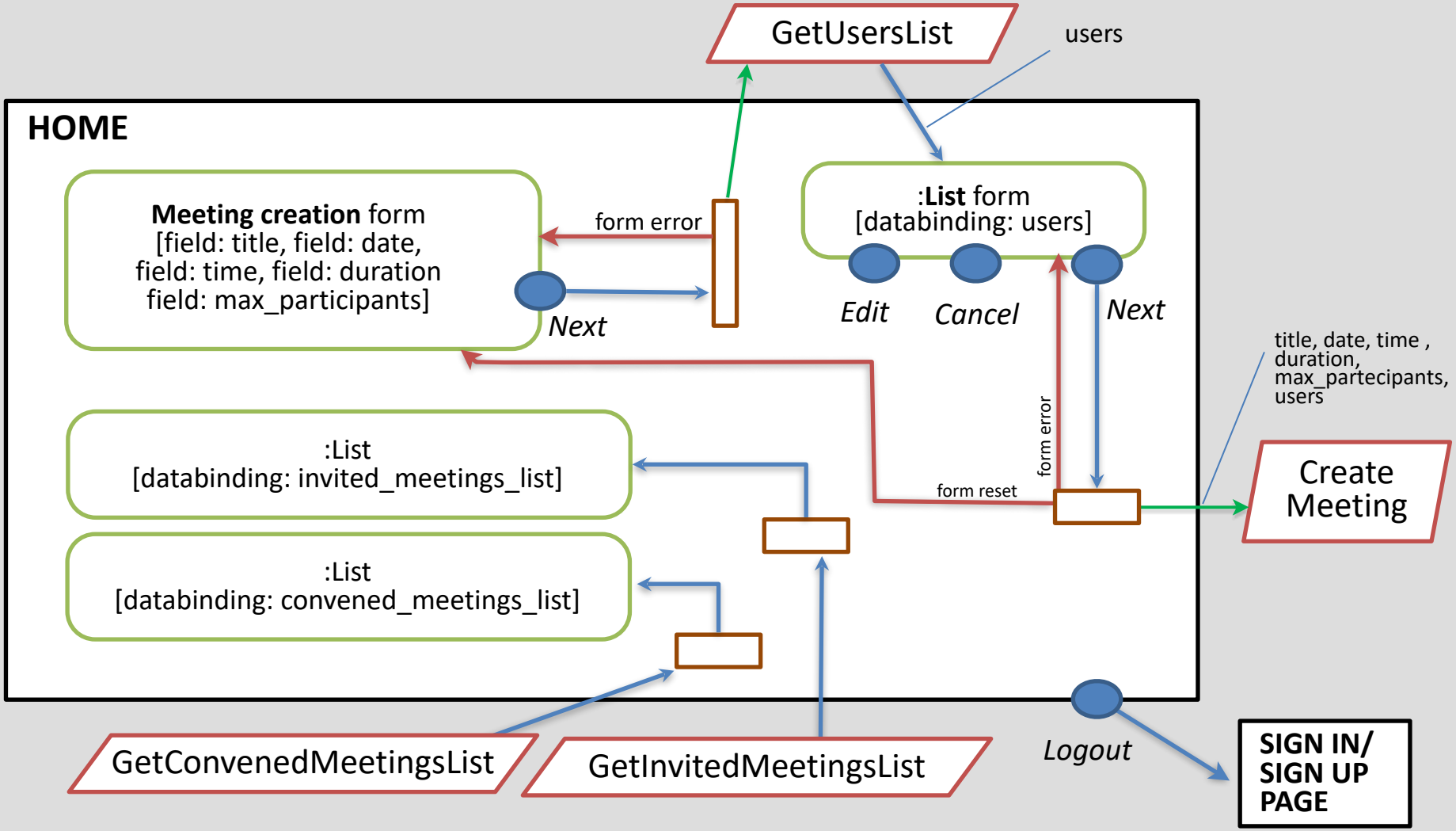
Pages (views), **view components**, **events**, **actions**

Completamento delle **specifiche**

- Tutti dati sono obbligatori.
- Deve essere selezionato almeno un invitato.
- Le riunioni non possono essere create per date nel passato.
- Il numero di partecipanti massimo deve essere positivo e non nullo.
- Gli utenti possono effettuare in qualsiasi momento il logout.

Application **design**




GetUsersList
users
HOME
Meeting creation form
[field: title, field: date,
field: time, field: duration
field: max_participants]
form error
Next
:List form
[databinding: users]
Edit
Cancel
Next
title, date, time ,
duration,
max_partecipants,
users
:List
[databinding: invited_meetings_list]
:List
[databinding: convened_meetings_list]
form error
form reset
Create
Meeting
GetConvenedMeetingsList
GetInvitedMeetingsList
Logout
SIGN IN/
SIGN UP
PAGE

Components

- **Model objects** (Beans)
  - User
  - Meeting
- **Data Access Objects** (Classes)
  - UserDAO
    - checkCredentials(String username, String password)
    - findAllUsersExcept(String username)
    - isAvailableUsername(String username)
    - createUser(User user)
  - MeetingDAO
    - createMeeting(Meeting meeting)
    - addInvitation(int meetingId, String guest)
    - findInvitedMeetingsByUser(String username)
    - findConvenedMeetingsByUser(String username)

Eventi

	Client side		Server side	
	Evento	**Azione**	**Evento**	**Azione**
INDEX	LoginComponent → Submit	Controllo dati inseriti	POST(credenziali)	Controllo correttezza credenziali
INDEX	RegisterComponent → Submit	Controllo unicità username, Controllo validità campi	POST(username)	Controllo unicità username
INDEX			POST(dati account)	Controllo validità campi, Creazione di un nuovo account
INDEX	ChangeView→ Click	Mostra/nascondi form di sign-in/sign-up	-	-
HOME	Load	Rendering dei dati sulle riunioni dell'utente	GET ()	Estrazione meeting indetti e meeting a cui l'utente è stato invitato
HOME	FormComponent → Next→ Click	Controllo validità campi, Cambio della vista	GET ()	Estrazione elenco utenti registrati
HOME	FormComponent → Edit → Click	Cambio della vista	-	-
HOME	FormComponent → Cancel→ Click	Reset form, Cambio della vista	-	-
HOME	FormComponent → Create→ Click	Controllo validità selezioni	POST (dati meeting)	Controllo validità campi, Aggiunta meeting e relativi inviti

Controllori

	Client side		Server side	
	Evento	**Azione**	**Evento**	**Controllore**
INDEX	LoginComponent → Submit	LoginComponent.doLogin()	POST(credenziali)	CheckCredentials
	RegisterComponent → Submit	RegisterComponent.createAccount()	POST(username)	CheckUsername
			POST(dati account)	CreateAccount
	ChangeView→ Click	PageManager.changeView()	-	-
HOME	Load	PageManager.start()	GET ()	GetConvenedMeetingsList, GetInvitedMeetingsList
	FormComponent → Next→ Click	FormComponent.getUsersList()	GET()	GetUsersList
	FormComponent → Edit → Click	FormComponent.changeStep()	-	-
	FormComponent → Cancel→ Click	FormComponent.reset()	-	-
	FormComponent → Create→ Click	makeCall	POST (dati meeting)	CreateMeeting

View

- **Index**
  - **LoginComponent:** gestisce la procedura di verifica delle credenziali
    - *load*(): associa al componente le funzioni per gestirne gli eventi
    - *doLogin*(): richiede al server la verifica delle credenziali
    - *reset*(): effettua il reset del form
    - *clearError*(),*setError*(msg): gestiscono la visualizzazione degli errori

  - **RegisterComponent:** gestisce la procedura di registrazione alla piattaforma
    - *load*(): associa al componente le funzioni per gestirne gli eventi
    - *createAccount*(): richiede al server la creazione dell'account
    - *reset*(): effettua il reset del form
    - *clearError*(),*setError*(msg): gestiscono la visualizzazione degli errori

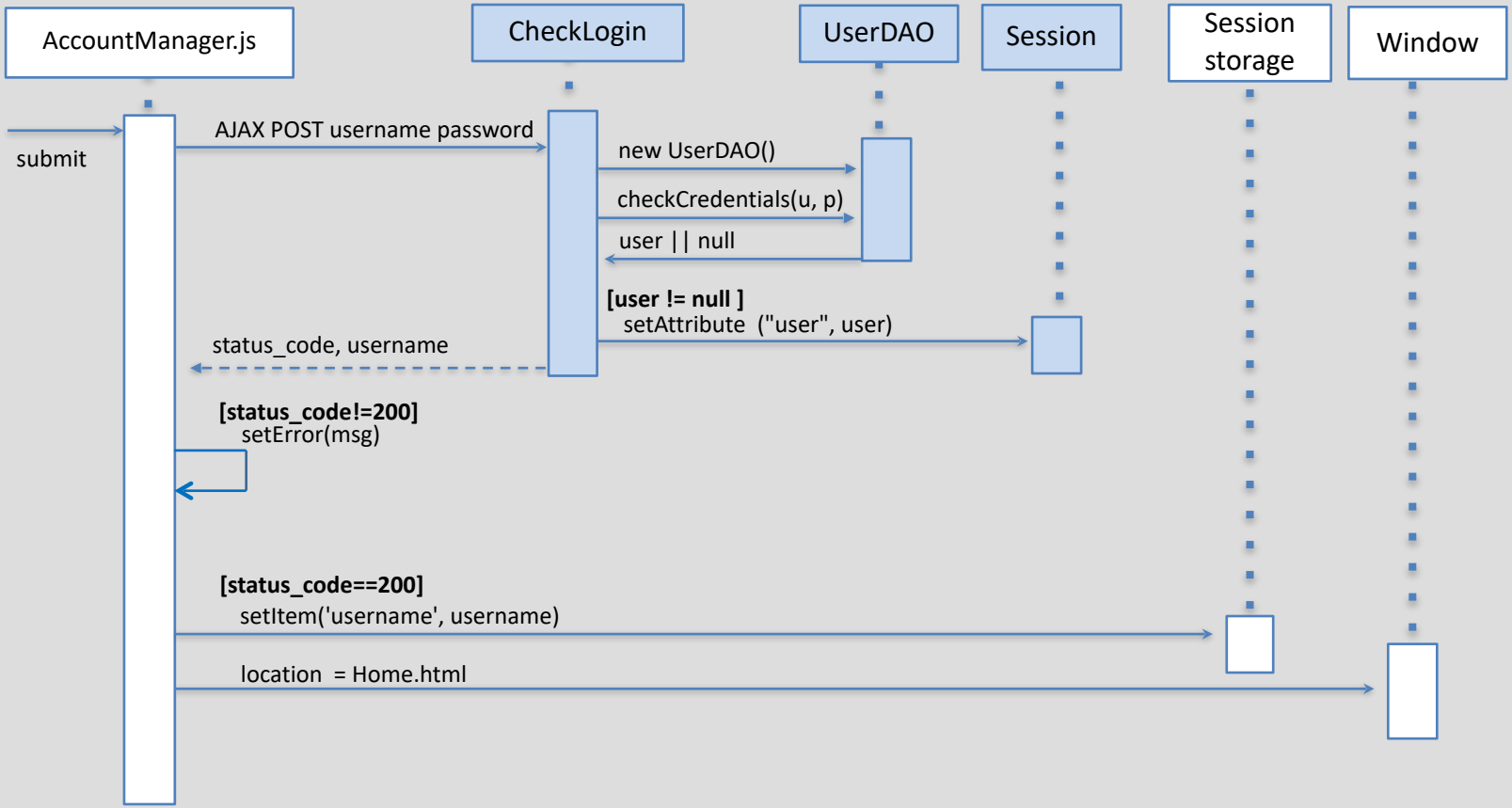
View

- **Home**
  - **InvitedMeetingsList:** visualizza tutte le riunioni alle quali l'utente è stato invitato
    - *reset*(): rimuove tutte le informazioni
    - *show*(): richiede al server le informazioni sulle riunioni non scadute
    - *update*(meetings): mostra le informazioni relative alle riunioni

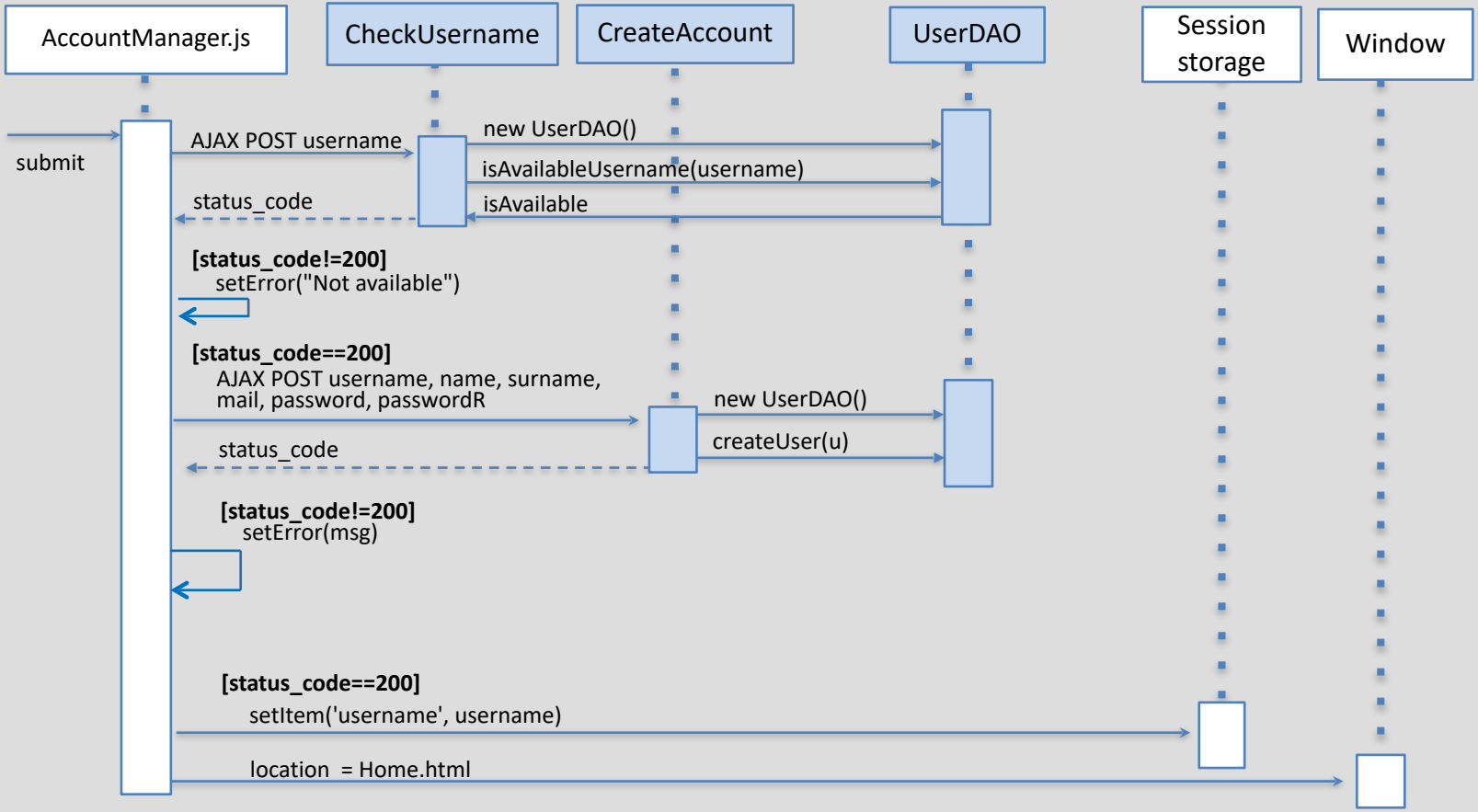
  - **ConvenedMeetingsList:** visualizza tutte le riunioni organizzate dall'utente
    - *reset*(): rimuove tutte le informazioni
    - *show*(): richiede al server le informazioni sulle riunioni non scadute
    - *update*(meetings): mostra le informazioni relative alle riunioni

  - **FormComponent:** consente la creazione di una nuova riunione
    - *load*(): associa al componente le funzioni per gestirne gli eventi
    - *getUsersList*(): richiede al server la lista degli utenti della piattaforma
    - *updateUsersList*(users): mostra la lista degli utenti che possono essere invitati
    - *changeStep*(): cambia il modulo visualizzato
    - *reset*(): ripristina il form allo stato iniziale

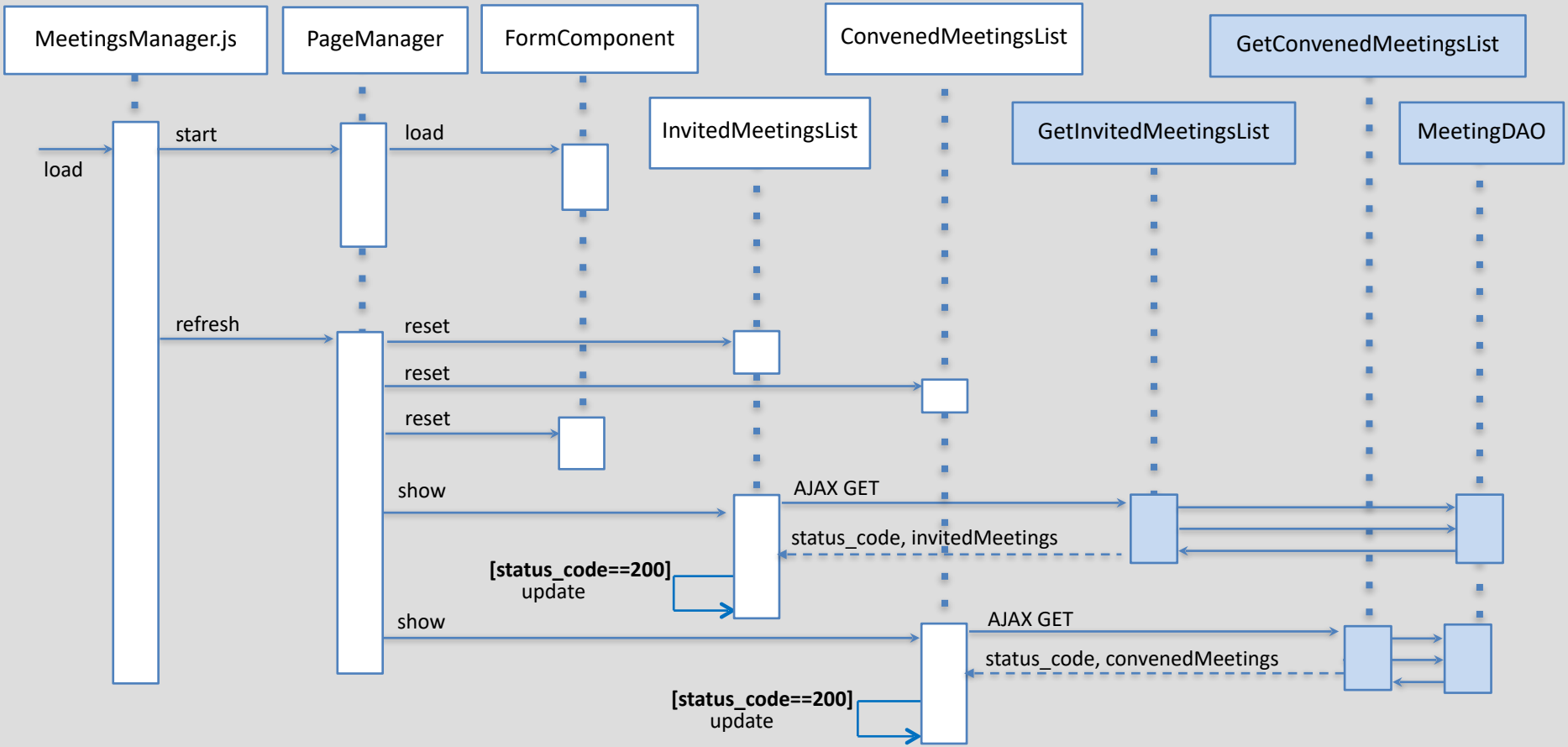
Event: **Sign-in**



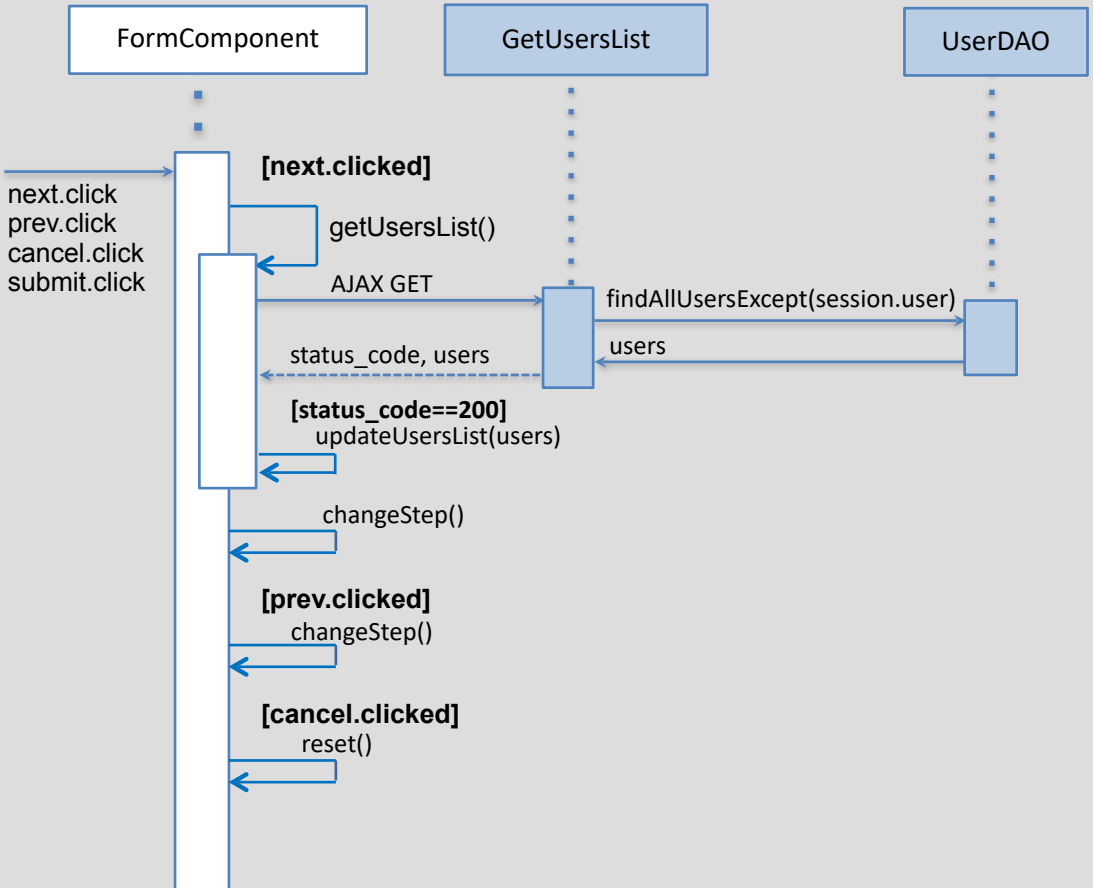
Event: **Sign-up**

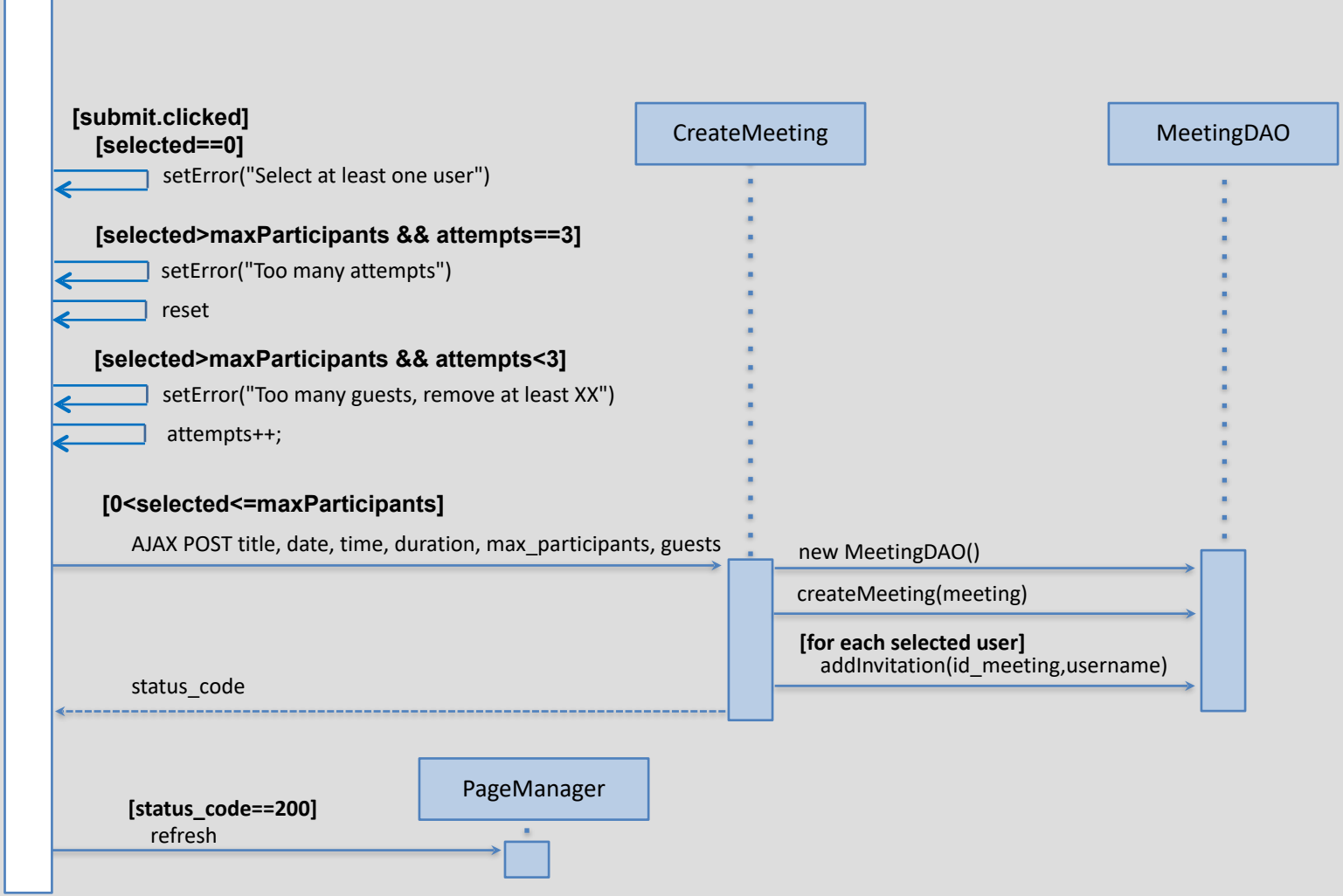


Event: **Load Home**



Event: **Create Meeting**




[submit.clicked]
[selected==0]
setError("Select at least one user")
[selected>maxParticipants && attempts==3]
setError("Too many attempts")
reset
[selected>maxParticipants && attempts<3]
setError("Too many guests, remove at least XX")
attempts++;
[0<selected<=maxParticipants]
AJAX POST title, date, time, duration, max_participants, guests
CreateMeeting
MeetingDAO
new MeetingDAO()
createMeeting(meeting)
[for each selected user]
addInvitation(id_meeting,username)
status_code
PageManager
[status_code==200]
refresh

Event: **Logout**

POST
/Logout

From:
Home.html