Progetto di **Tecnologie Informatiche per il Web**

Gruppo 86
Salim Salici
Matricola: 10640001
Cod. persona: 893196

POLITECNICO
MILANO 1863

Exercise 3: Gruppi di lavoro

- **Pure HTML** version

Un’applicazione web consente la gestione di gruppi di lavoro. 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”.
La registrazione controlla l’unicità dello username.

Un gruppo ha un titolo, una data di creazione, una durata dell’attività in giorni e un numero minimo e massimo di partecipanti. L’utente fa il login e, se autenticato, accede all’HOME page che mostra l’elenco dei gruppi creati da lui e ancora attivi, l’elenco delle gruppi cui è stato invitato e ancora attivi, e una form per creare un nuovo gruppo.

Quando l’utente seleziona un gruppo compare una pagina DETTAGLI GRUPPO che mostra i dati del gruppo e la lista degli utenti partecipanti identificati da nome e cognome. Quando l’utente inoltra la form di creazione di un nuovo gruppo con il bottone INVIA, appare una pagina ANAGRAFICA con l’elenco degli utenti registrati identificati da nome e cognome e ordinata alfabeticamente per cognome crescente. L’utente può scegliere uno o più partecipanti dall’elenco e premere il bottone INVITA per invitarli al gruppo. Se il numero d’invitati è inferiore di X unità rispetto al minimo ammissibile, appare di nuovo la pagina ANAGRAFICA con un messaggio “Troppo pochi utenti selezionati, aggiungerne almeno X”. Gli utenti precedentemente selezionati devono rimanere selezionati, ma possono essere sostituiti, in tutto o in parte, da altri utenti. Se il numero d’invitati è superiore di X unità rispetto al massimo ammissibile, appare di nuovo la pagina ANAGRAFICA con un messaggio “Troppi utenti selezionati, eliminarne almeno X”. In questo caso, la pagina evidenzia nell’elenco gli utenti scelti in precedenza come preselezionati, in modo che l’utente possa deselezionarne alcuni. Se alla pressione del bottone INVITA il numero d’invitati rispetta i vincoli, il gruppo è memorizzato nella base di dati e associato agli utenti invitati e l’utente è rimandato alla HOME PAGE. Al terzo tentativo scorretto di assegnare un numero errato di invitati a un gruppo appare una pagina CANCELLAZIONE con un messaggio “Tre tentativi di definire un gruppo con un numero di partecipanti errato, il gruppo non sarà creato” e un link per tornare all’HOME PAGE. In questo caso il gruppo NON è memorizzato nella base di dati. L’applicazione non deve registrare nella base di dati gruppi con numero errato di partecipanti. L’applicazione consente il logout dell’utente.

Exercise 3: Gruppi di lavoro

- **JavaScript [RIA]** version

Si realizzi un'applicazione client-server web che modifica le specifiche precedenti 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 dell'utente, 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.
- Nella pagina DETTAGLI GRUPPO se l'utente è il creatore del gruppo appare l'icona di un cestino. L'utente può trascinare il nome e cognome di un partecipante sul cestino per cancellarne la partecipazione. A seguito del rilascio sul cestino l'applicazione controlla che sia rispettato il numero minimo di partecipanti. Se il vincolo è rispettato l'operazione è eseguita nella base di dati e un messaggio di conferma viene emesso. Se il vincolo NON è rispettato l'operazione NON è eseguita nella base di dati e un messaggio di diniego viene emesso.
- La scelta dall'anagrafica deve essere realizzata con una pagina modale con i bottoni invia e cancella. NB: una pagina modale è una finestra che si apre per dare una qualche scelta o un qualche messaggio all'utente: https://it.wikipedia.org/wiki/Finestra_modale.
- 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.

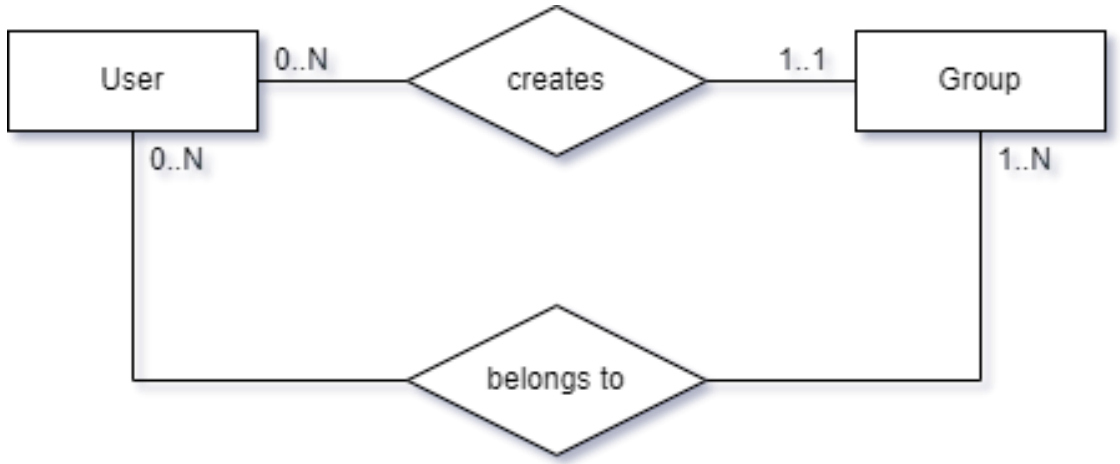
Database

– Un’applicazione web consente la gestione di gruppi di lavoro. 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”.
La registrazione controlla l’unicità dello **username**.

Un **gruppo** ha un **titolo**, una **data di creazione**, una **durata** dell’attività in giorni e un **numero minimo e massimo di partecipanti**. L’**utente** fa il login e, se autenticato, accede all’HOME page che mostra l’elenco dei **gruppi creati da lui** e ancora attivi, l’elenco delle **gruppi cui è stato invitato** e ancora attivi, e una form per creare un nuovo gruppo.

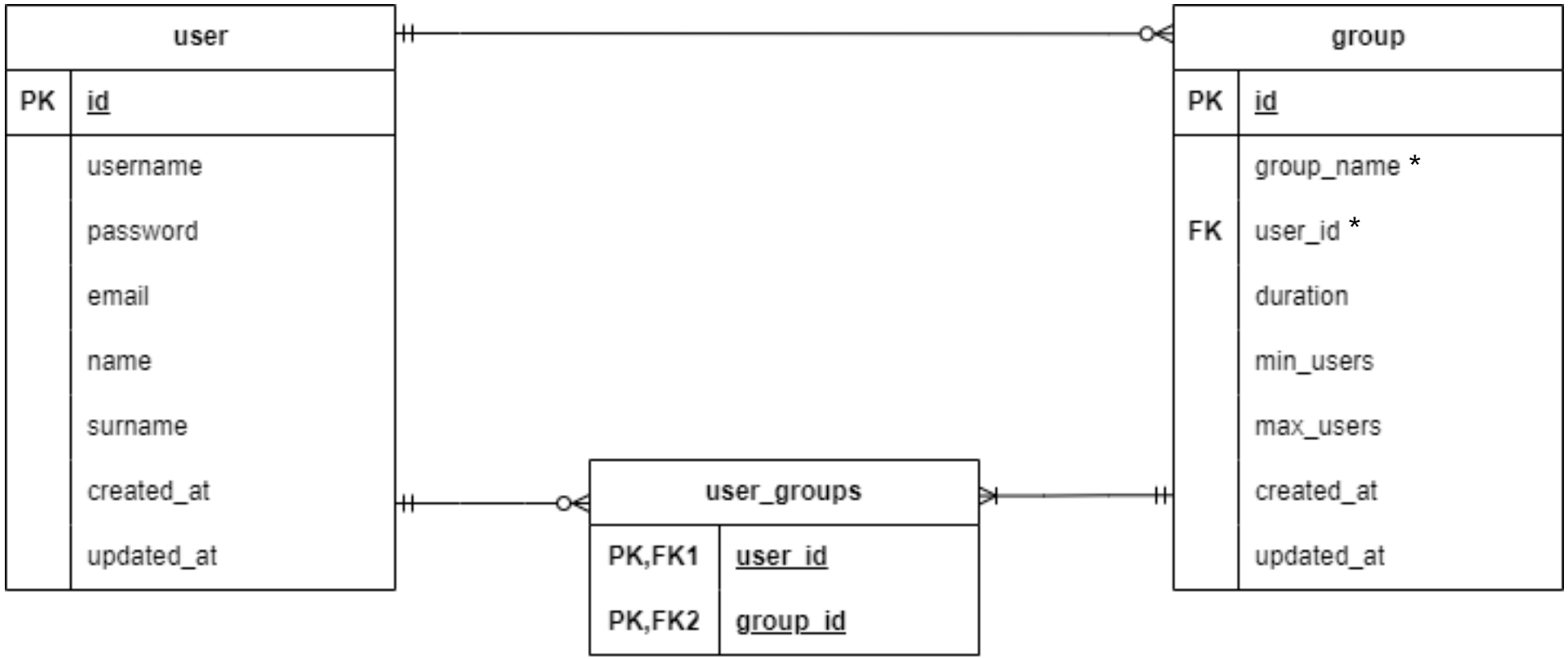
Quando l’utente seleziona un gruppo compare una pagina DETTAGLI GRUPPO che mostra i dati del gruppo e la lista degli utenti partecipanti identificati da **nome** e **cognome**. [...]

ER schema design



● Entity ● Attribute ● Association

Logical schema design



* group_name and user_id together form a unique key constraint (a user cannot create multiple groups with the same name)

Data Definition Language

Users

```
CREATE TABLE 'users' (
  'id' int NOT NULL AUTO_INCREMENT,
  'username' varchar(45) NOT NULL,
  'email' varchar(100) NOT NULL,
  'password' varchar(255) NOT NULL,
  'name' varchar(45) NOT NULL,
  'surname' varchar(45) NOT NULL,
  'created_at' datetime NOT NULL
            DEFAULT CURRENT_TIMESTAMP,
  'updated_at' datetime
            DEFAULT NULL
            ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY ('id'),
  UNIQUE KEY 'username' ('username')
)
```

Groups

```
CREATE TABLE 'groups' (
  'id' int NOT NULL AUTO_INCREMENT,
  'group_name' varchar(100) NOT NULL,
  'user_id' int NOT NULL,
  'duration' int NOT NULL,
  'min_users' int NOT NULL,
  'max_users' int NOT NULL,
  'created_at' datetime NOT NULL
            DEFAULT CURRENT_TIMESTAMP,
  'updated_at' datetime NOT NULL
            DEFAULT CURRENT_TIMESTAMP
            ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY ('id'),
  UNIQUE KEY 'group_name' ('group_name','user_id'),
  KEY 'user_fk_idx' ('user_id'),
  CONSTRAINT 'user_fk' FOREIGN KEY ('user_id')
            REFERENCES 'users' ('id') ON UPDATE CASCADE
)
```

Users and Groups association

```
CREATE TABLE 'user_groups' (
  'user_id' int NOT NULL,
  'group_id' int NOT NULL,
  PRIMARY KEY ('user_id','group_id'),
  KEY 'user_groups_group_id_fk_idx' ('group_id'),
  CONSTRAINT 'user_groups_group_id_fk'
            FOREIGN KEY ('group_id') REFERENCES 'groups' ('id')
            ON UPDATE CASCADE,
  CONSTRAINT 'user_groups_user_id_fk'
            FOREIGN KEY ('user_id') REFERENCES 'users' ('id')
            ON UPDATE CASCADE
)
```

Requisiti dell’applicazione

Un’applicazione web consente la gestione di gruppi di lavoro. 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”. La registrazione controlla l’unicità dello username.

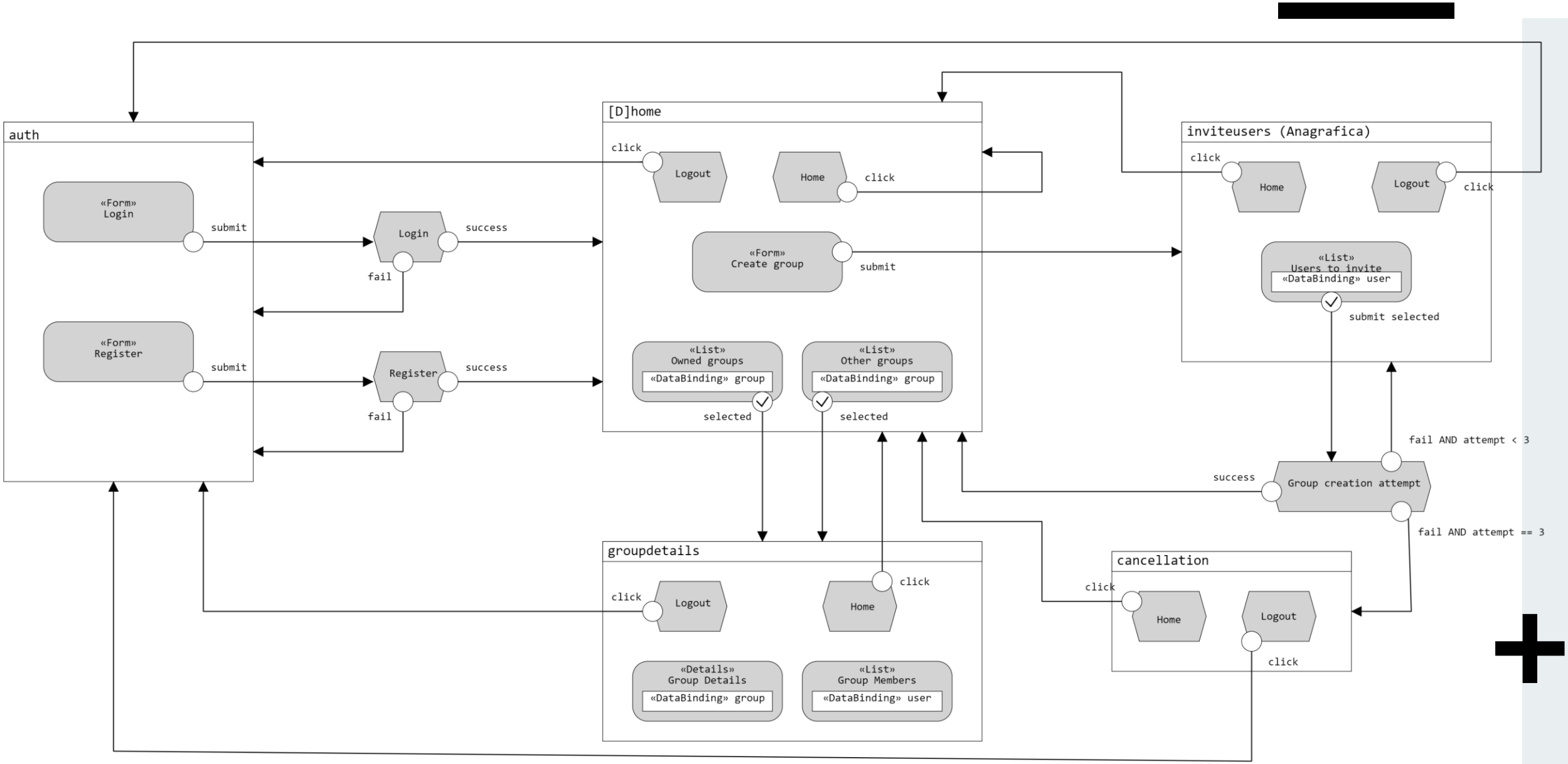
Un gruppo ha un titolo, una data di creazione, una durata dell’attività in giorni e un numero minimo e massimo di partecipanti. L’utente fa il login e, se autenticato, accede all’**HOME page** che mostra l’elenco dei **gruppi creati da lui e ancora attivi**, **l’elenco delle gruppi cui è stato invitato e ancora attivi**, e una **form per creare un nuovo gruppo**.

Quando l’utente **seleziona un gruppo** compare una pagina **DETTAGLI GRUPPO** che mostra i **dati del gruppo** e la **lista degli utenti partecipanti** identificati da nome e cognome. Quando l’utente inoltra la form di creazione di un nuovo gruppo con il **bottone INVIA**, appare una pagina **ANAGRAFICA** con **l’elenco degli utenti registrati** identificati da nome e cognome e ordinata alfabeticamente per cognome crescente. **L’utente può scegliere uno o più partecipanti dall’elenco e premere** il **bottone INVITA** per **invitarli** al gruppo. Se il numero d’invitati è inferiore di X unità rispetto al minimo ammissibile, appare di nuovo la pagina ANAGRAFICA con un **messaggio** “Troppo pochi utenti selezionati, aggiungerne almeno X”. Gli utenti precedentemente selezionati devono rimanere selezionati, ma possono essere sostituiti, in tutto o in parte, da altri utenti. Se il numero d’invitati è superiore di X unità rispetto al massimo ammissibile, appare di nuovo la pagina ANAGRAFICA con un **messaggio** “Troppi utenti selezionati, eliminarne almeno X”. In questo caso, la pagina evidenzia nell’elenco gli utenti scelti in precedenza come preselezionati, in modo che l’utente possa **deselezionarne alcuni**. Se alla **pressione del bottone INVITA** il numero d’invitati rispetta i vincoli, il **gruppo è memorizzato nella base** di dati e associato agli utenti invitati e l’utente è rimandato alla HOME PAGE. Al **terzo tentativo scorretto** di assegnare un numero errato di invitati a un gruppo appare una pagina **CANCELLAZIONE** con un **messaggio** “Tre tentativi di definire un gruppo con un numero di partecipanti errato, il gruppo non sarà creato” e un **link** per tornare all’HOME PAGE. In questo caso il gruppo NON è memorizzato nella base di dati. L’applicazione non deve registrare nella base di dati gruppi con numero errato di partecipanti. L’applicazione consente il **logout** dell’utente.

● Pages ● View components ● Events ● Actions

PURE HTML version

Application design - IFML



Components

Model objects (Beans)
- Group
- User

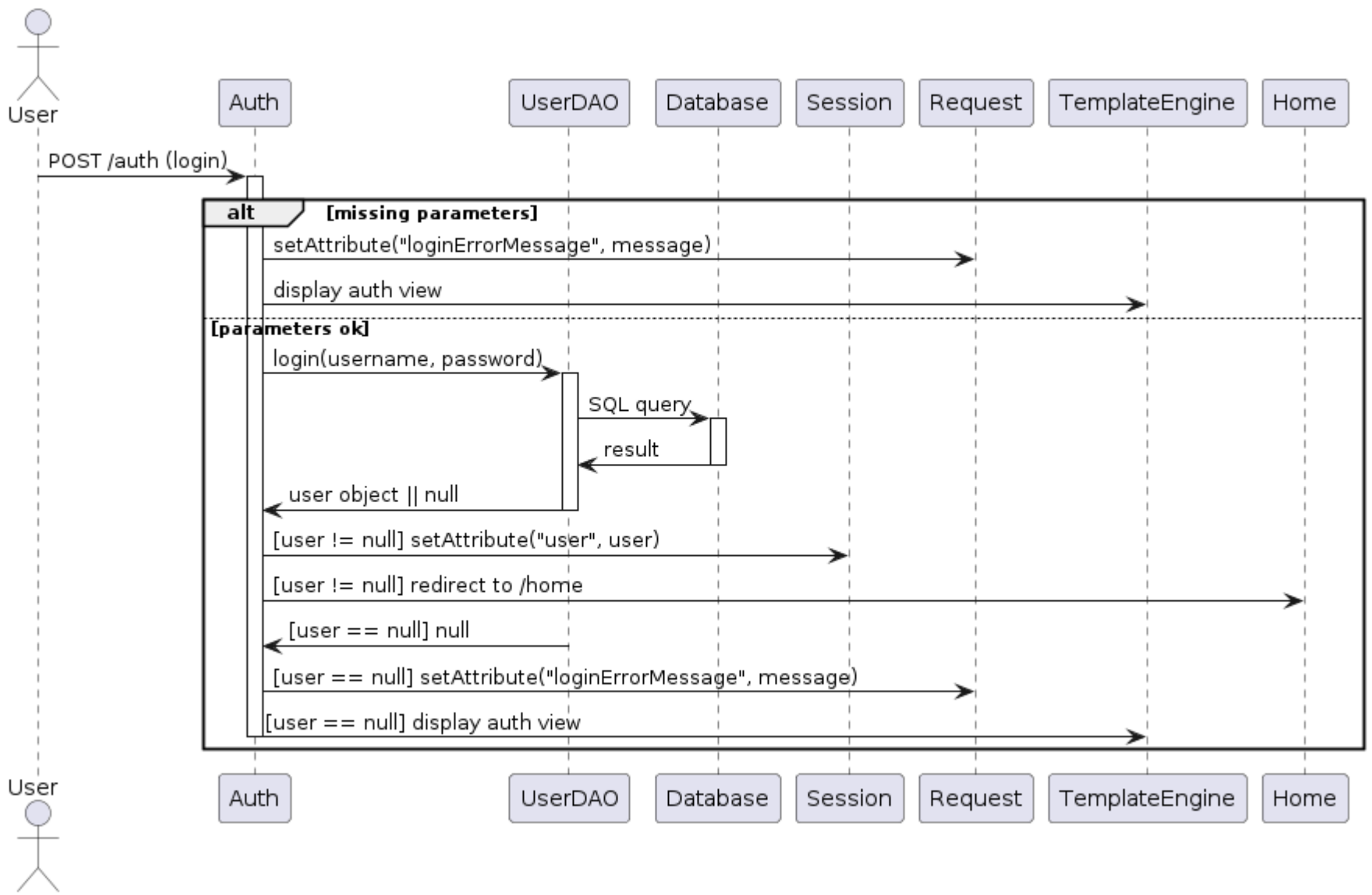
Data Access Objects
- GroupDAO
  - fetchGroupById(groupId)
  - fetchUsersOfGroup(groupId)
  - isGroupNameAvailableForUser(userId, groupName)
  - createNewGroup (name, creator_id, duration, minUsers, maxUsers, userIds)
- UserDAO
  - fetchAllUsers()
  - login(username, password)
  - isUsernameAvailable(username)
  - register(username, email, password, name, surname)
  - fetchGroupsOwnedBy(userId)
  - fetchGroupsWithUser(userId)

Filters
- UserLoggedFilter
- VisitorFilter
- ValidationCleanupFilter

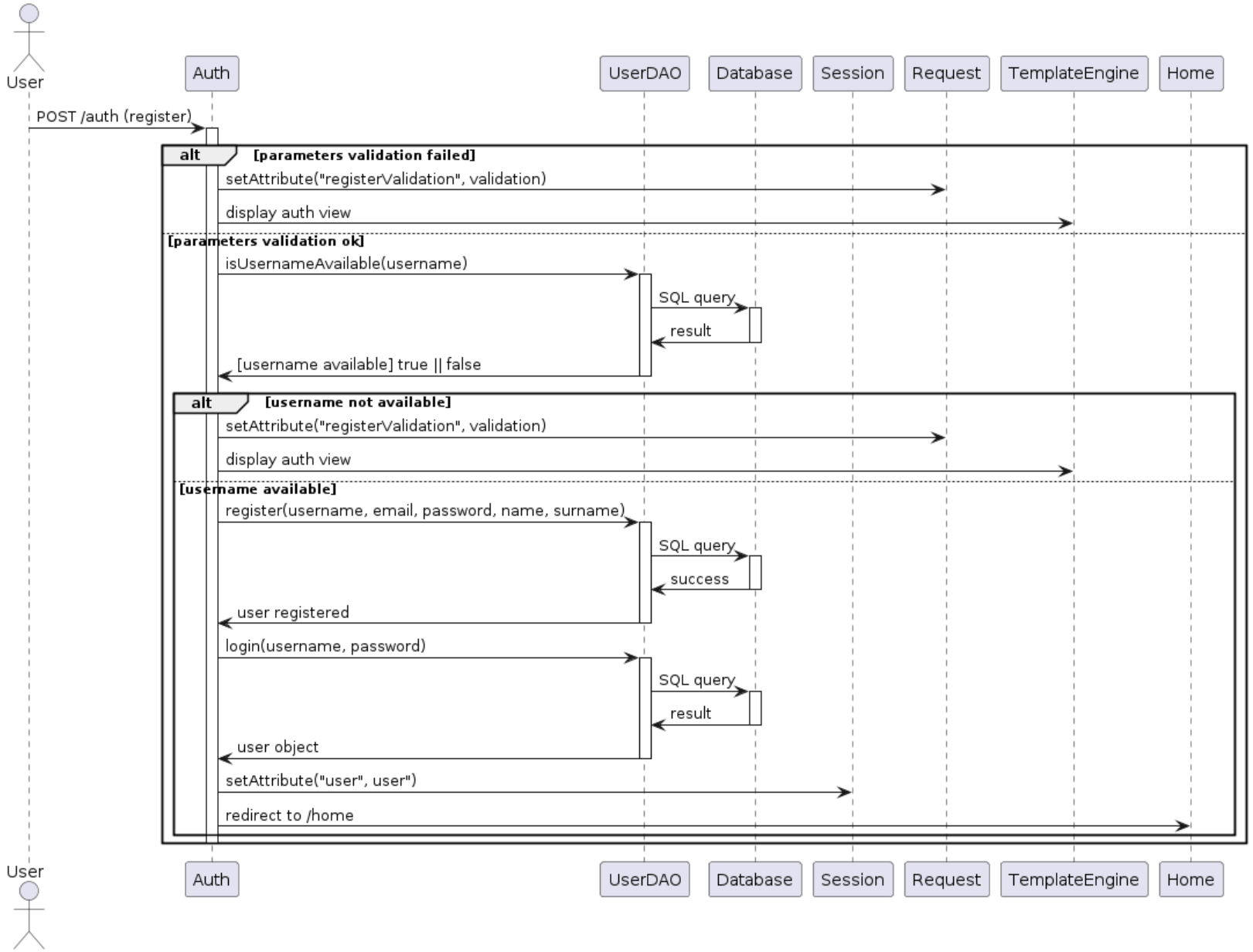
Controllers (Servlets)
- Auth
- Home
- GroupDetails
- InviteUsers
- Cancellation
- Error
- Logout

Views
- auth
- header (fragment)
- home
- groupdetails
- inviteusers (anagrafica)
- cancellation
- error

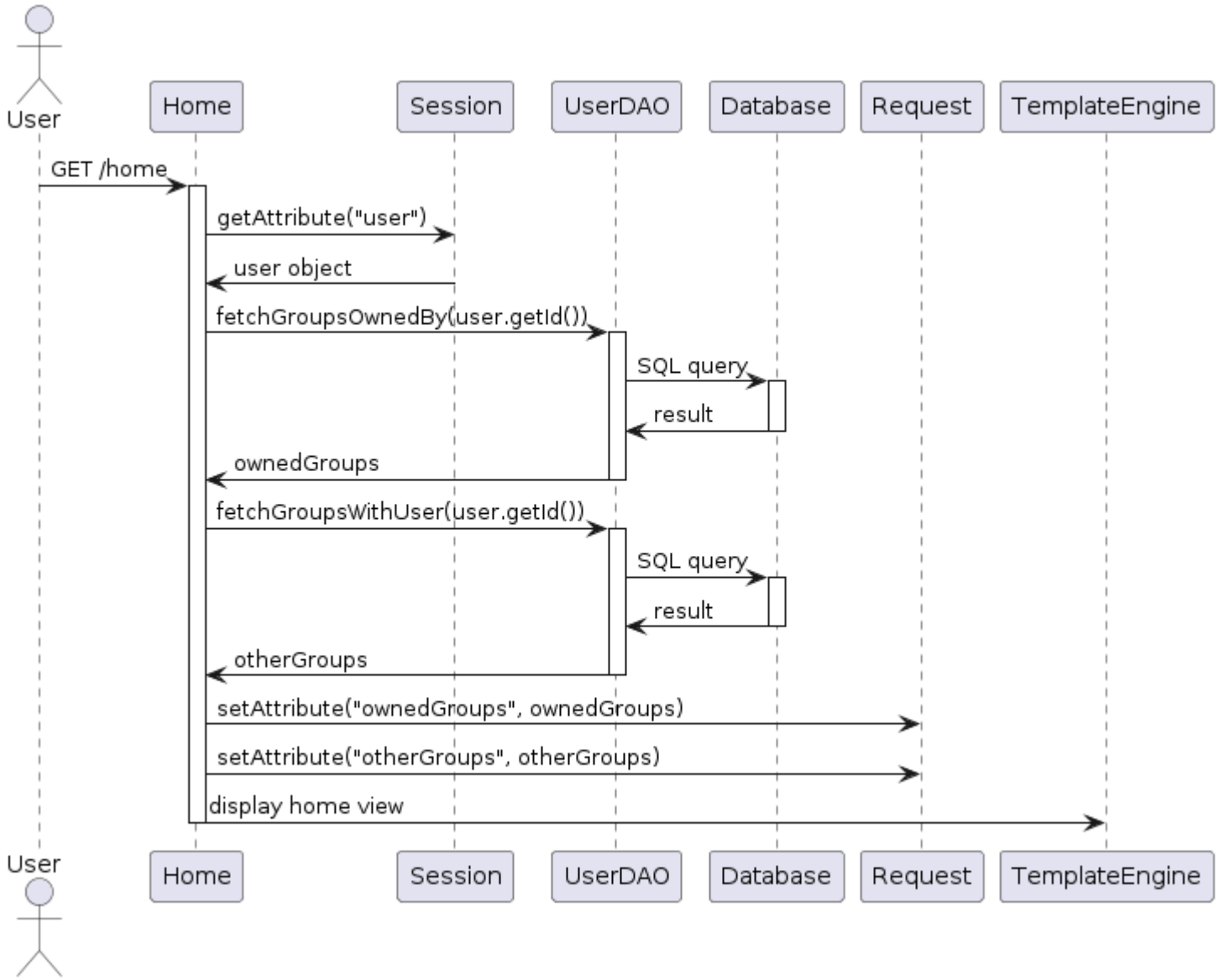
Login sequence diagram



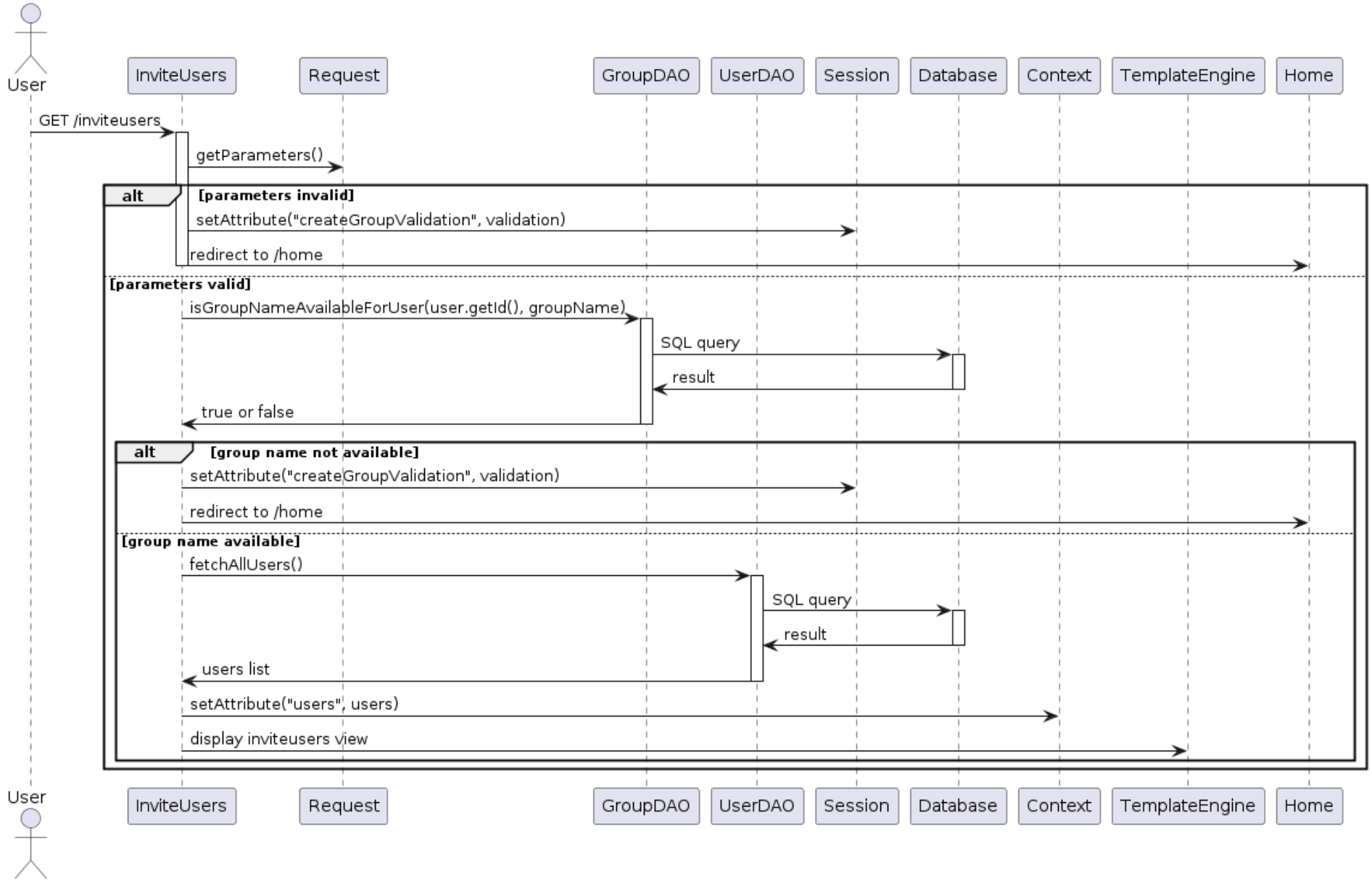
Registration sequence diagram



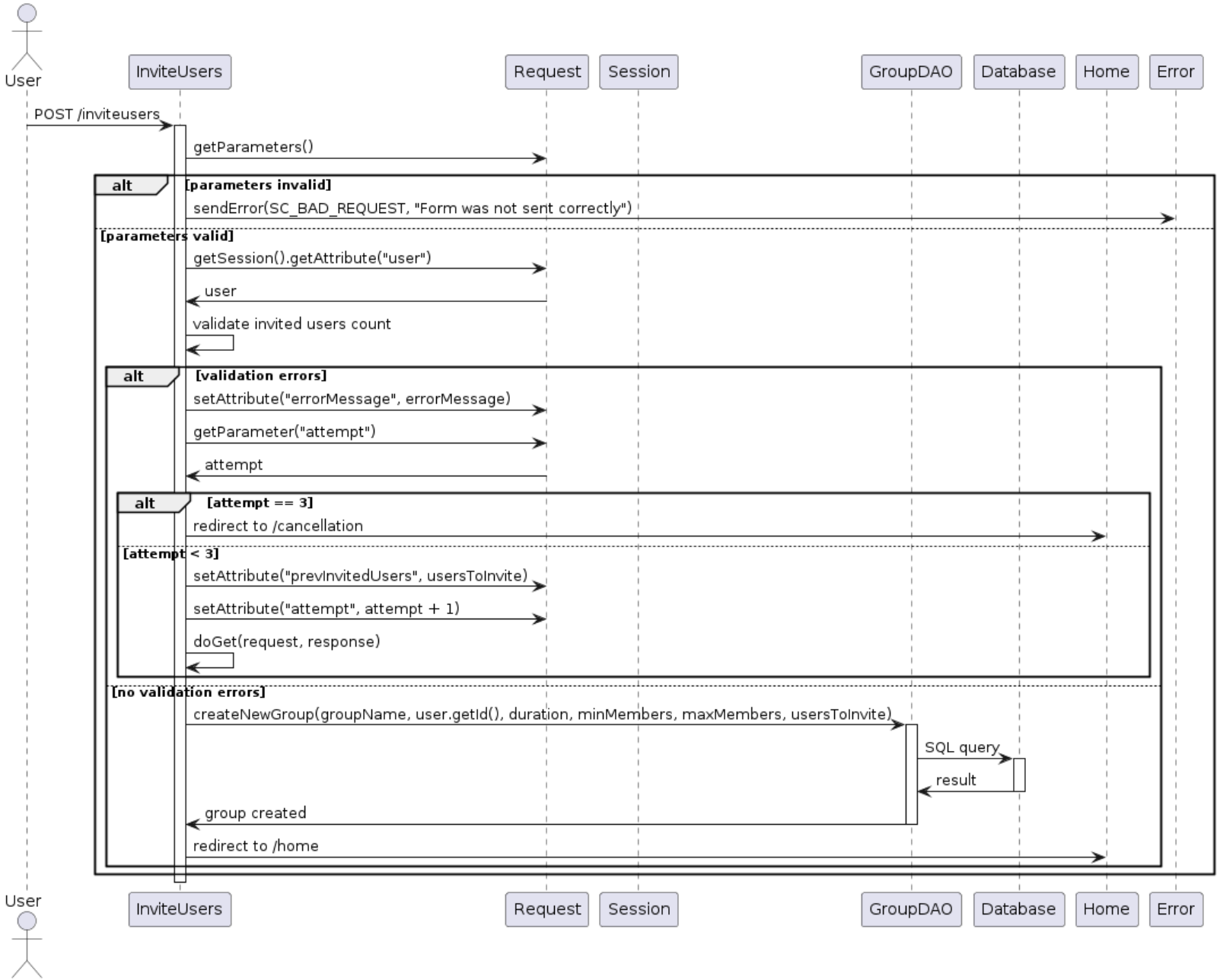
Home sequence diagram



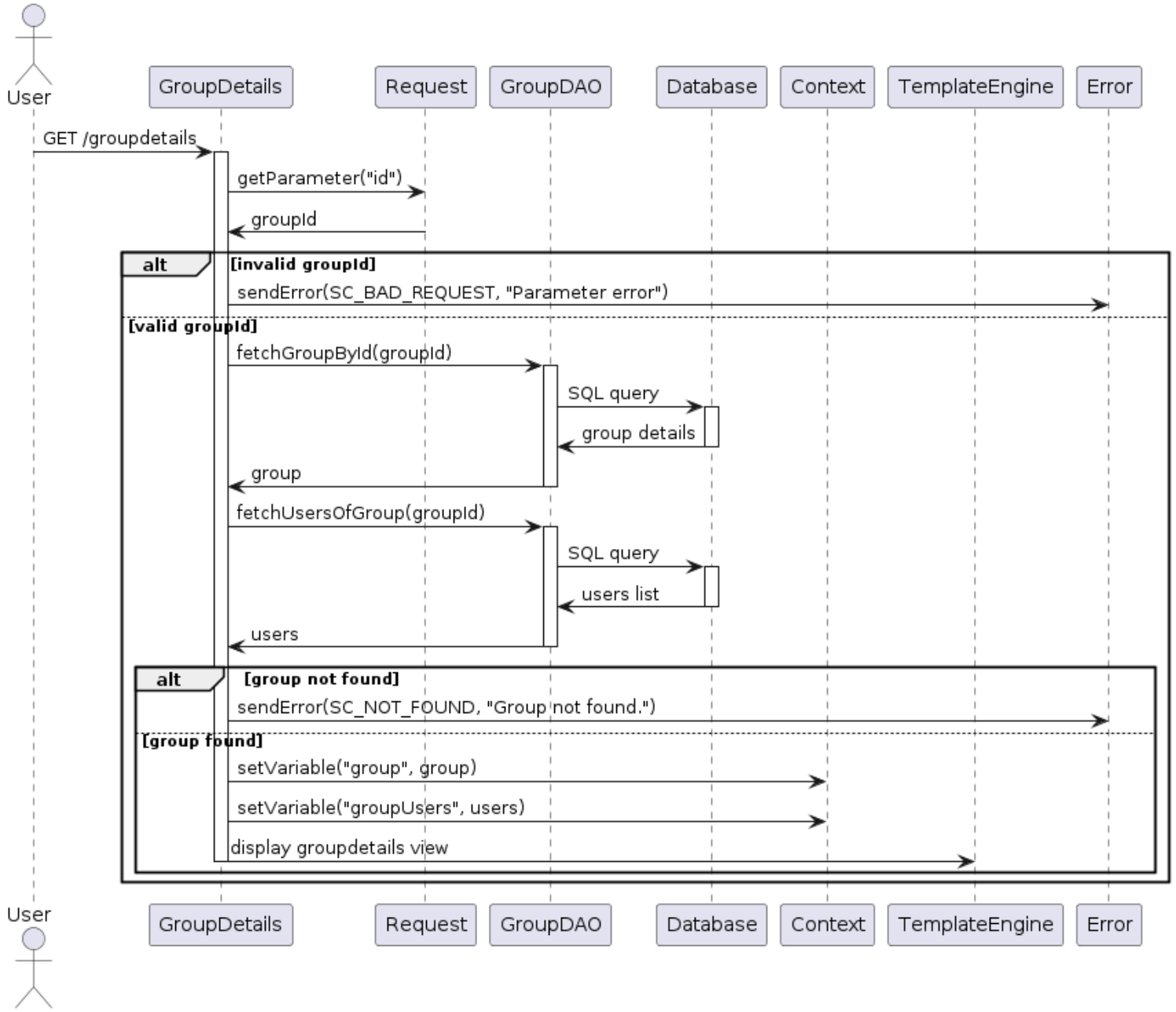
InviteUsers [GET] sequence diagram



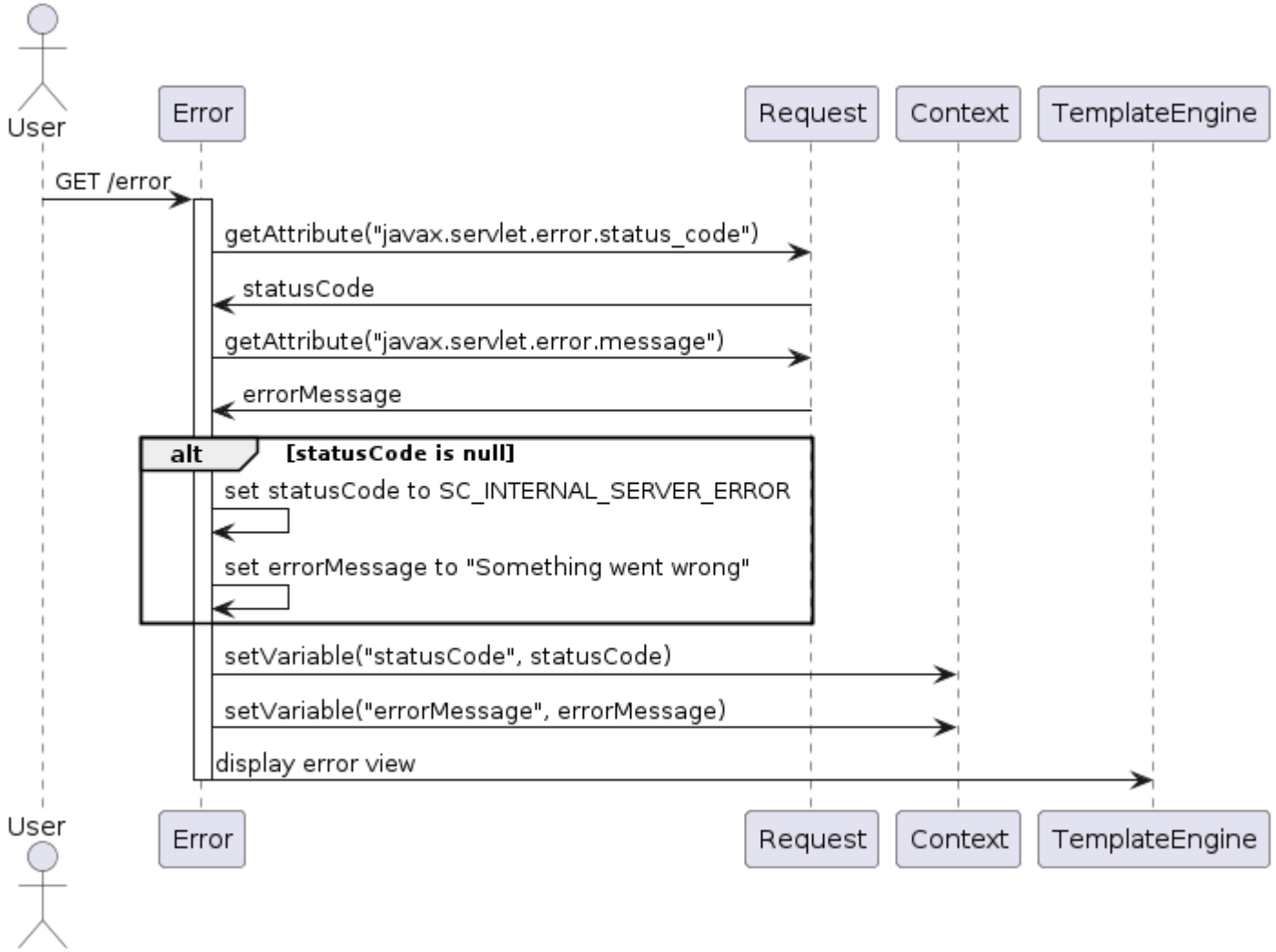
InviteUsers [POST] sequence diagram



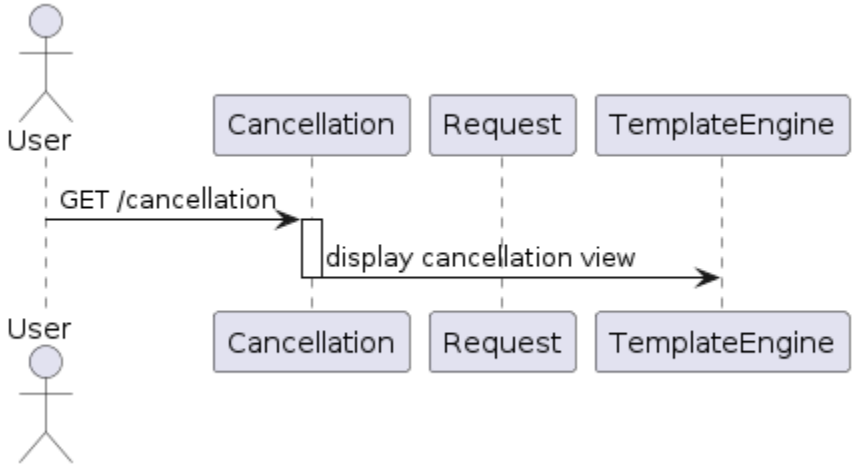
GroupDetails sequence diagram



Error sequence diagram

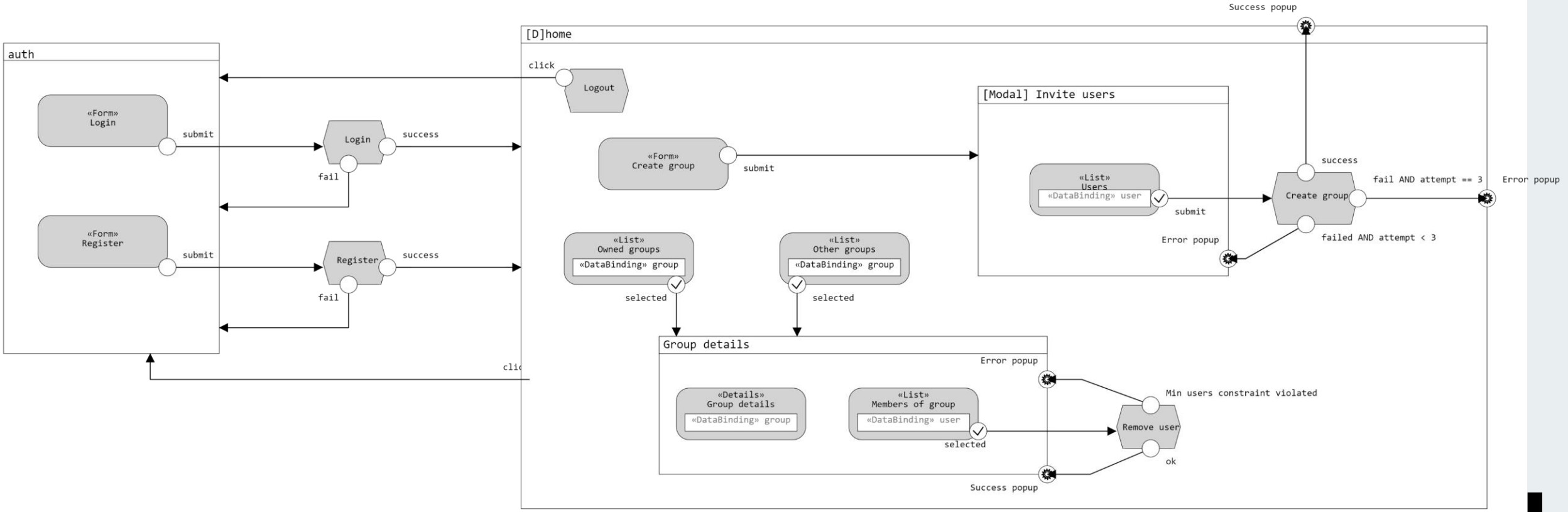


Cancellation sequence diagram



RIA version

Application design - IFML



Server side components

Model objects (Beans)
- Group
- User

Data Access Objects
- GroupDAO
  - fetchGroupById(groupId)
  - fetchUsersOfGroup(groupId)
  - isGroupNameAvailableForUser(userId, groupName)
  - createNewGroup (name, creator_id, duration, minUsers, maxUsers, userIds)
  - removeMemberFromGroup(groupId, memberId)
- UserDAO
  - fetchAllUsers()
  - login(username, password)
  - isUsernameAvailable(username)
  - register(username, email, password, name, surname)
  - fetchGroupsOwnedBy(userId)
  - fetchGroupsWithUser(userId)

Controllers (Servlets)
- LoginCheck
- RegisterCheck
- Logout
- OwnedActiveGroups
- ActiveGroups
- GroupDetails
- Users
- RemoveGroupMember

Filters
- UserLoggedFilter

Client side components

Auth - *auth.html*

- **Login form**
  - Handles validation and submit
- **Register form**
  - Handles validation and submit

Home - *home.html*

- **CreateGroupManager**: Component that handles the creation of a new group, from specifying details to the modal window for selecting which users to invite.
  - **registerEvents()**: Associates event handlers for managing the group creation process.
  - **displayUserSelection()**: Populates the modal with the list of users that the group creator can invite to the new group.
  - **displayErrors()**: Displays error messages if there are issues during group creation.
  - **hideErrors()**: Hides any error messages.
  - **openModal()**: Opens the modal window to select users to invite to the group.
  - **closeModal()**: Closes the modal window.

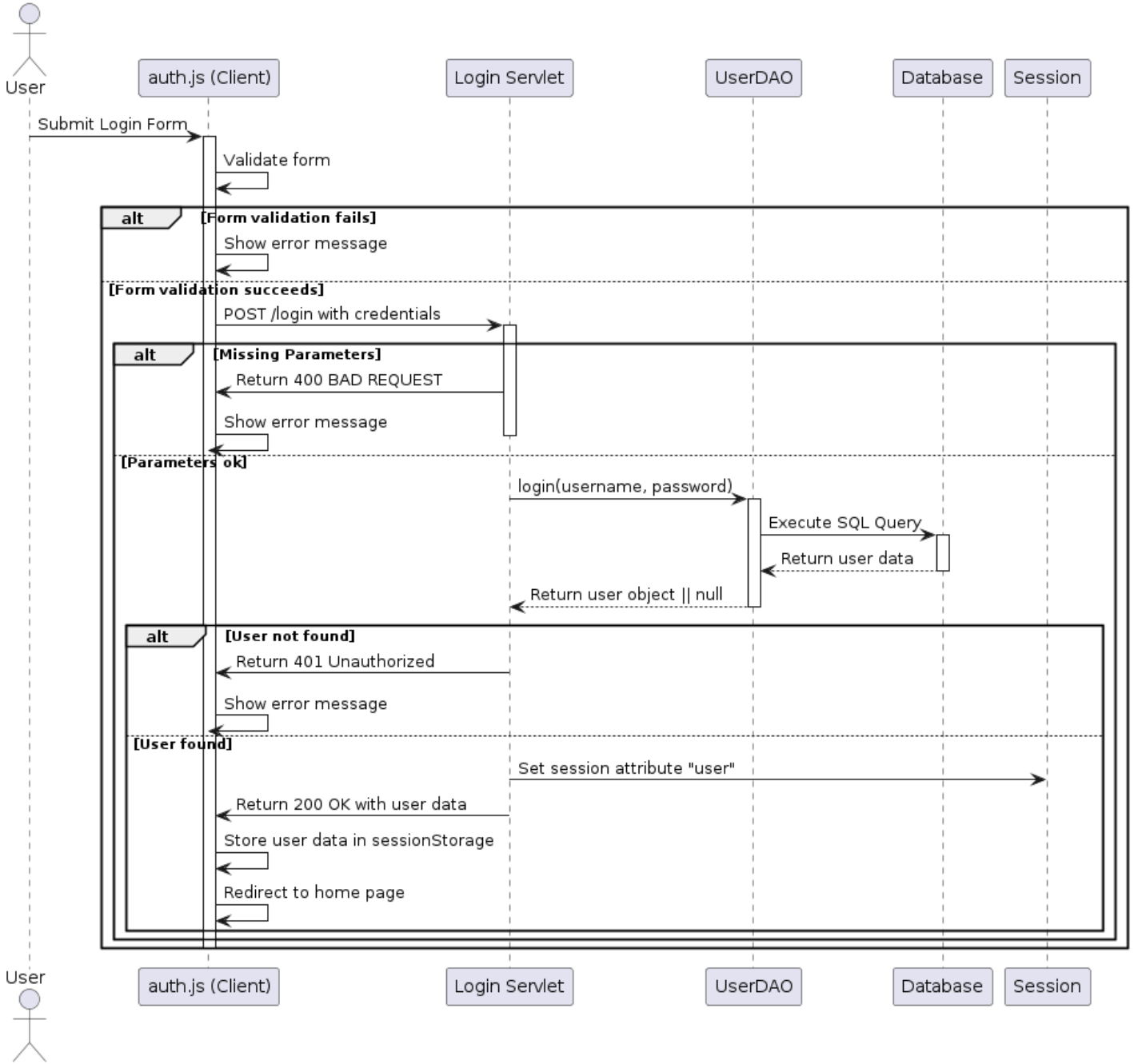
- **GroupsManager**: Component responsible for managing and displaying the preview of the groups that the user owns or belongs to.
  - **fetchAndDisplayGroups()**: Requests the list of groups from the server and then displays them.
  - **displayGroups()**: Displays the groups obtained from the server.
  - **removeMemberFromGroup()**: Manages the removal of a specific member from a group, updating the display based on changes.

- **GroupsDetails**: Component responsible for managing and displaying the details of groups that the user owns or belongs to.
  - **fetchAndExpand()**: Requests information about the group from the server and displays it.
  - **expand()**: Expands the group container to show information and members.
  - **setupTrash()**: Called only if the user is the owner of the group; sets up the element responsible for removing group members via drag and drop.
  - **removeUser()**: Removes the element displaying a group member after they are removed.

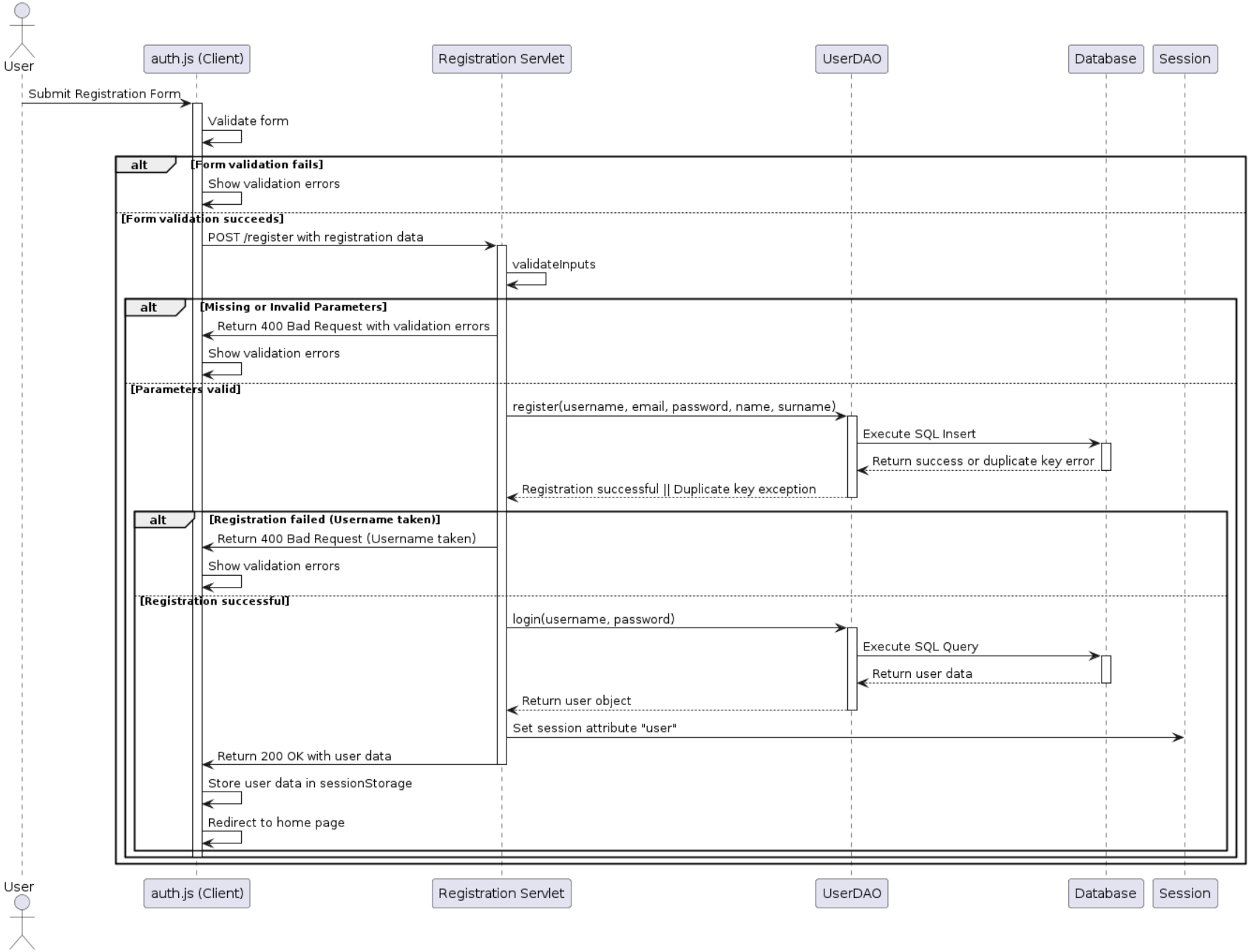
Client Controller – Event handling

Evento lato client	Controller lato client	Evento generato lato server	Controller Lato server
auth → login form → submit	function makeCall	POST username, password	LoginCheck (servlet)
auth → register form → submit	function makeCall	POST username, email, name, surname, password, confirm password	RegisterCheck (servlet)
home → logout → click	function logout	GET	Logout (servlet)
home → create group form → click «Select users to invite» button	CreateGroupManager → validation → makeCall → displayUserSelection()	GET	Users (servlet)
home → create group form → submit	CreateGroupManager → validation → makeCall → GroupsManager → fetchAndDisplayGroups()	POST [group details], [list of members]	CreateGroup (servlet)
home → create group form → modal → click «Cancel» button	CreateGroupManager → form.reset() → closeModal()	-	-
home → group → click «View details» button	GroupDetails → makeCall	GET groupId	GroupDetails (servlet)
home → group → user → dragstart	GroupDetails → [anon func]	-	-
home → group → user → dragend	GroupDetails → [anon func]	-	-
home → group → trash → drop	GroupDetails → makeCall → removeGroupMemeber()	POST groupId, memberId	RemoveGroupMember (servlet)

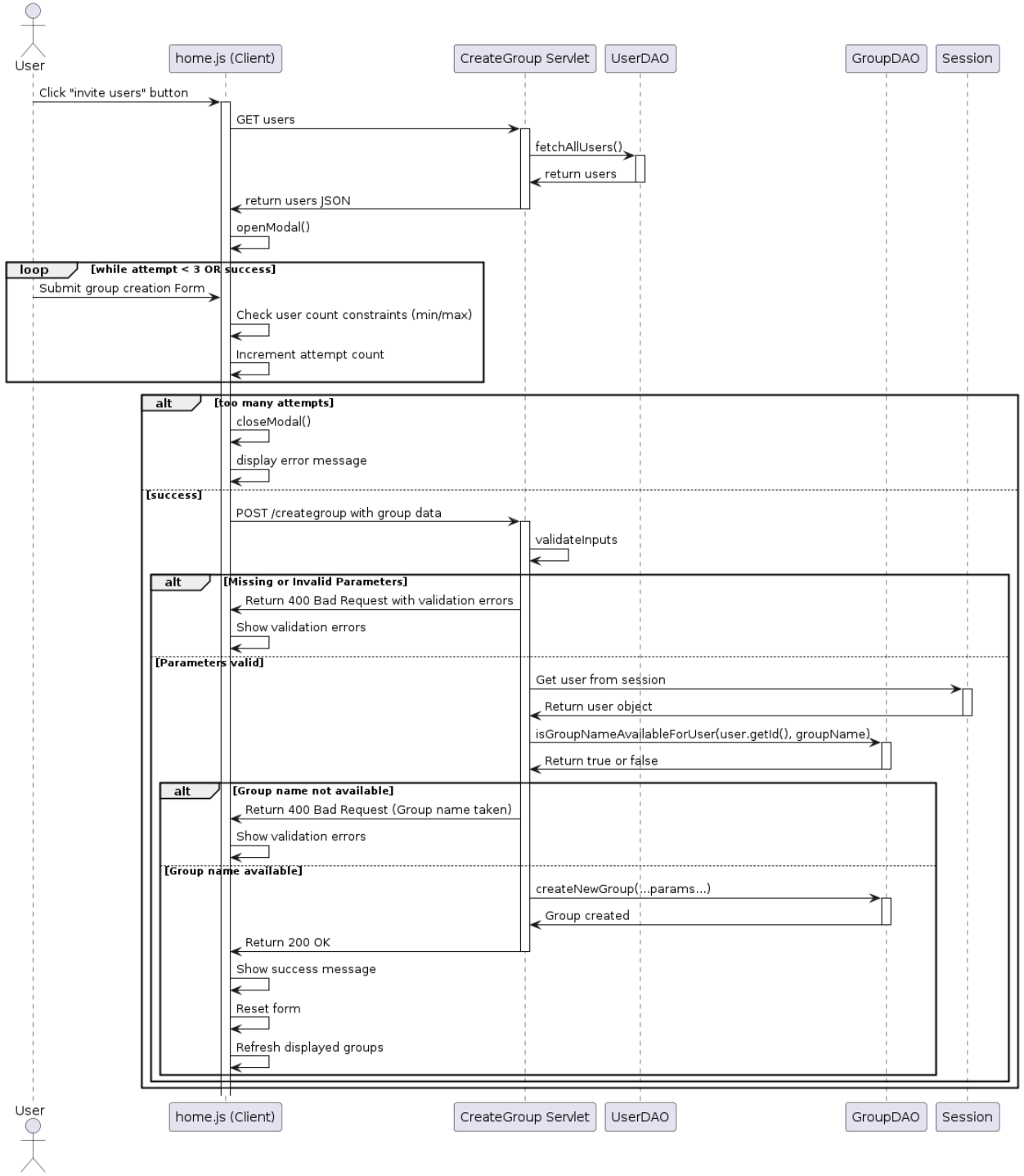
Login


User
auth.js (Client)
Login Servlet
UserDAO
Database
Session
Submit Login Form
Validate form
alt
[Form validation fails]
Show error message
[Form validation succeeds]
POST /login with credentials
alt
[Missing Parameters]
Return 400 BAD REQUEST
Show error message
[Parameters ok]
login(username, password)
Execute SQL Query
Return user data
Return user object || null
alt
[User not found]
Return 401 Unauthorized
Show error message
[User found]
Set session attribute "user"
Return 200 OK with user data
Store user data in sessionStorage
Redirect to home page
User
auth.js (Client)
Login Servlet
UserDAO
Database
Session

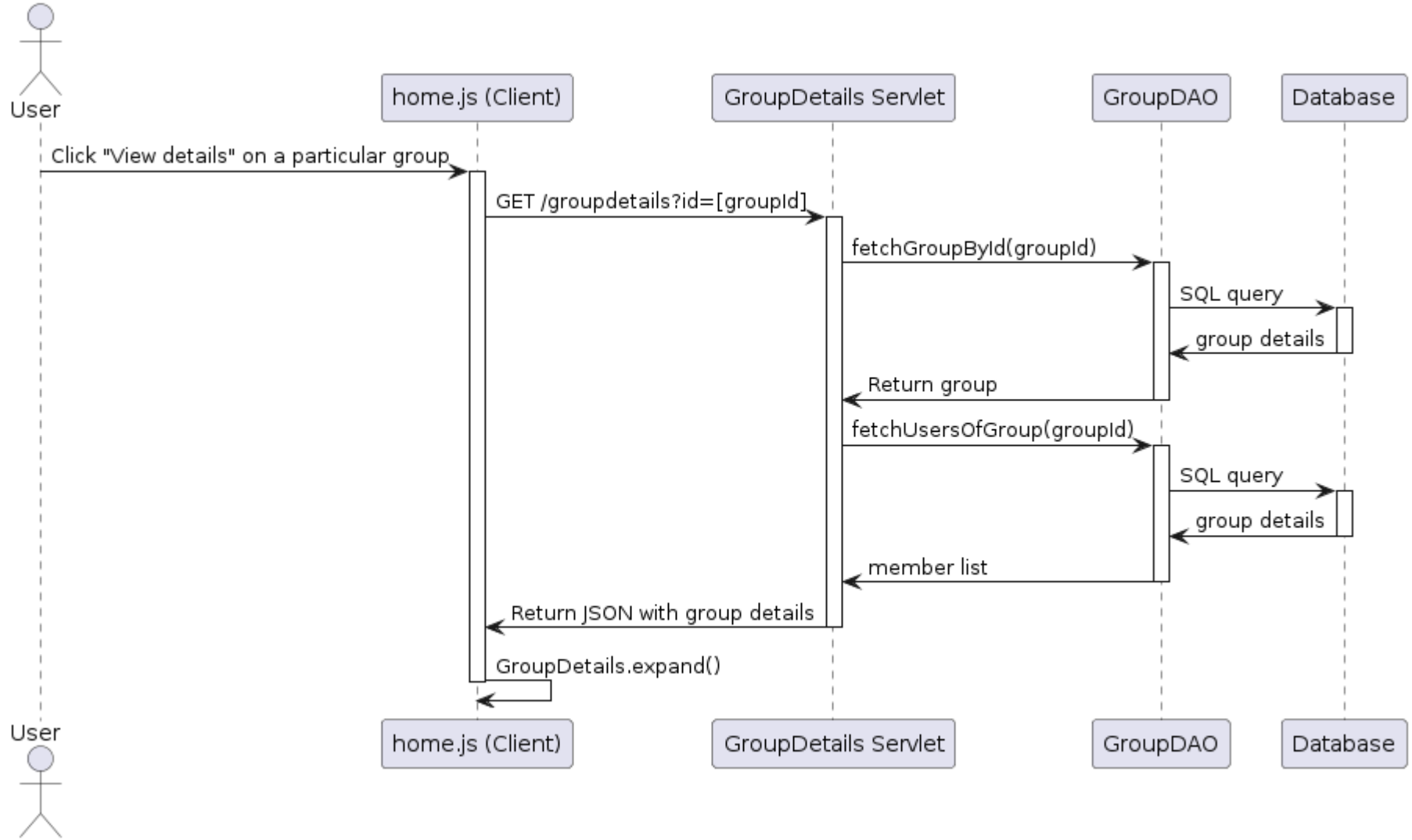
Registration



Create group



View details (Expands a group box to show its information and members)



Remove group Member

(with drag and drop)