Esercizio 4 - pure HTML

Giovanni Manfredi – Sebastiano Meneghin

Esercizio 4: trasferimento denaro

Un'applicazione web consente la gestione di trasferimenti di denaro online da un conto a un altro. 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 utente ha un nome, un cognome, uno username e uno o più conti correnti. Un conto ha un codice, un saldo, e i trasferimenti fatti (in uscita) e ricevuti (in ingresso) dal conto. Un trasferimento ha una data, un importo, un conto di origine e un conto di destinazione. Quando l'utente accede all'applicazione appare una pagina LOGIN per la verifica delle credenziali. In seguito all'autenticazione dell'utente appare l'HOME page che mostra l'elenco dei suoi conti. Quando l'utente seleziona un conto, appare una pagina STATO DEL CONTO che mostra i dettagli del conto e la lista dei movimenti in entrata e in uscita, ordinati per data discendente. La pagina contiene anche una form per ordinare un trasferimento. La form contiene i campi: codice utente destinatario, codice conto destinatario, causale e importo. All'invio della form con il bottone INVIA, l'applicazione controlla che il conto di destinazione appartenga all'utente specificato e che il conto origine abbia un saldo superiore o uguale all'importo del trasferimento. In caso di mancanza di anche solo una condizione, l'applicazione mostra una pagina con un avviso di fallimento che spiega il motivo del mancato trasferimento. Nel caso in cui entrambe le condizioni siano soddisfatte, l'applicazione deduce l'importo dal conto di origine, aggiunge l'importo al conto di destinazione e mostra una pagina CONFERMA TRASFERIMENTO che presenta i dati dell'importo trasferito e i dati del conto di origine e di destinazione con i rispettivi saldi precedenti al trasferimento e aggiornati dopo il trasferimento. L'applicazione deve garantire l'atomicità del trasferimento: ogni volta che il conto di destinazione viene addebitato, il conto di origine deve essere accreditato. Ogni pagina contiene un collegamento per tornare alla pagina precedente. L'applicazione consente il logout dell'utente.

Analisi dati per database

Un'applicazione web consente la gestione di trasferimenti di denaro online da un conto a un altro. 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 utente ha un nome, un cognome, uno username e uno o più conti correnti. Un conto ha un codice, un saldo, e i trasferimenti fatti (in uscita) e ricevuti (in ingresso) dal conto. Un trasferimento ha una data, un importo, un conto di origine e un conto di destinazione. Quando l'utente accede all'applicazione appare una pagina LOGIN per la verifica delle credenziali. In seguito all'autenticazione dell'utente appare l'HOME page che mostra l'elenco dei suoi conti. Quando l'utente seleziona un conto, appare una pagina STATO DEL CONTO che mostra i dettagli del conto e la lista dei movimenti in entrata e in uscita, ordinati per data discendente. La pagina contiene anche una form per ordinare un trasferimento. La form contiene i campi: codice utente destinatario, codice conto destinatario, causale e importo. All'invio della form con il bottone INVIA, l'applicazione controlla che il conto di destinazione appartenga all'utente specificato e che il conto origine abbia un saldo superiore o uguale all'importo del trasferimento. In caso di mancanza di anche solo una condizione, l'applicazione mostra una pagina con un avviso di fallimento che spiega il motivo del mancato trasferimento. Nel caso in cui entrambe le condizioni siano soddisfatte, l'applicazione deduce l'importo dal conto di origine, aggiunge l'importo al conto di destinazione e mostra una pagina CONFERMA TRASFERIMENTO che presenta i dati dell'importo trasferito e i dati del conto di origine e di destinazione con i rispettivi saldi precedenti al trasferimento e aggiornati dopo il trasferimento. L'applicazione deve garantire l'atomicità del trasferimento: ogni volta che il conto di destinazione viene addebitato, il conto di origine deve essere accreditato. Ogni pagina contiene un collegamento per tornare alla pagina precedente. L'applicazione consente il logout dell'utente.

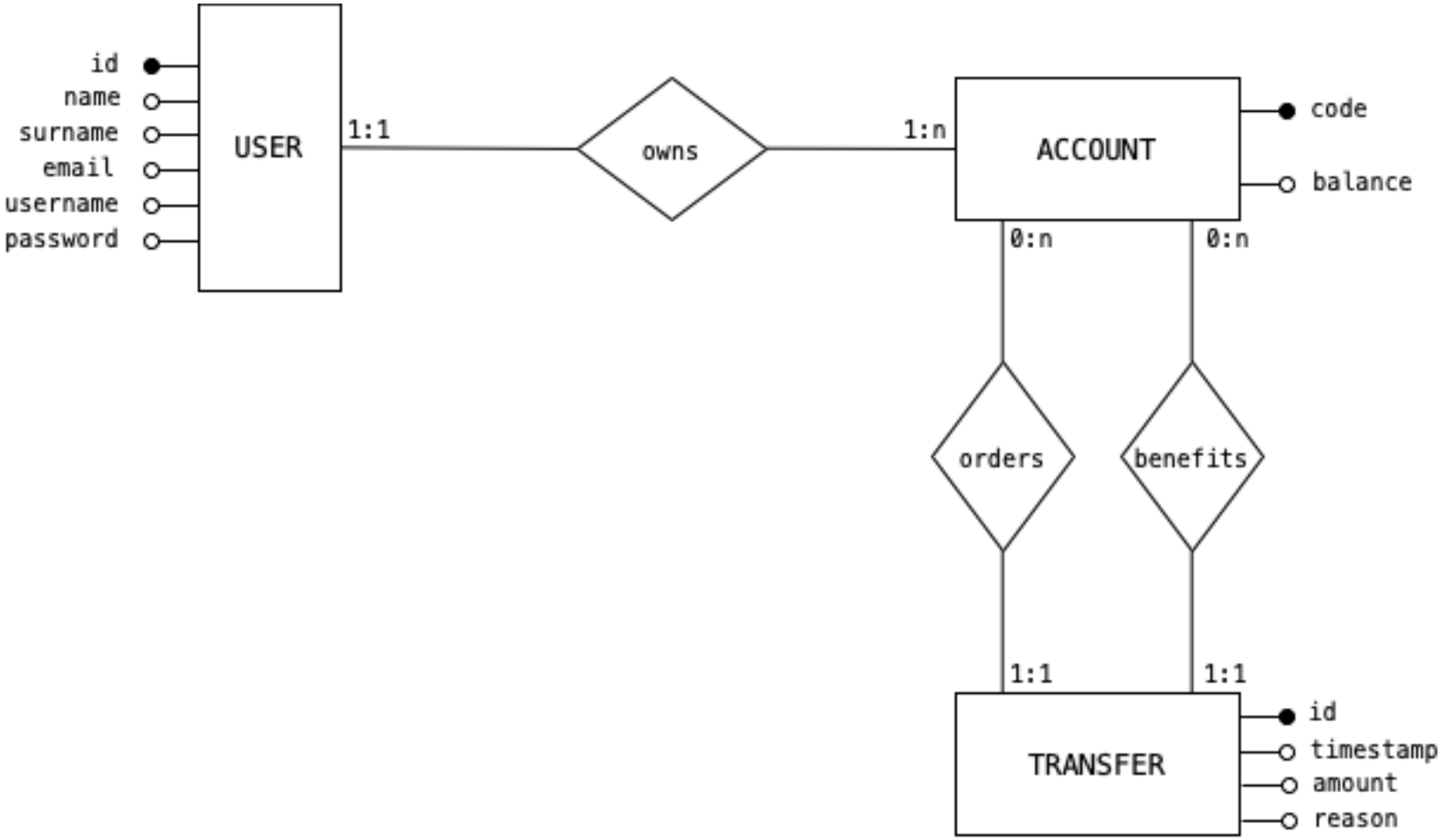
Entità, attributi, relazioni

Completamento delle specifiche (1/2) - database

- Il campo *email* è necessariamente univoco. Utenti diversi non possono avere la stessa email
- Il campo *username* e *email* sono distinti e entrambi univoci. Vanno specificati dall'utente nella registrazione
- Il saldo (*balance*) di ogni conto (*account*) deve essere in qualsiasi momento maggiore o uguale a zero
- L'importo (*amount*) di un trasferimento (*transfer*) deve essere sempre maggiore di zero
- Il conto di origine (*code_account_orderer*) di un trasferimento (*transfer*) non può corrispondere al conto di destinazione (*code_account_beneficiary*) del trasferimento
- La data (*timestamp*) del trasferimento non può essere anteriore al momento in cui questo viene ordinato

Entità, attributi, relazioni

Database design



Database schema (1/2)

```
CREATE TABLE `user` (
  `id` int unsigned NOT NULL AUTO_INCREMENT,
  `name`     varchar(45) NOT NULL,
  `surname`  varchar(45) NOT NULL,
  `email`    varchar(320) NOT NULL,
  `username` varchar(45) NOT NULL,
  `password` varchar(45) NOT NULL,
  PRIMARY KEY (`id`),
  CONSTRAINT `user_username_un` UNIQUE
(`username`),
  CONSTRAINT `user_email_un` UNIQUE (`email`)
) ENGINE=InnoDB
```

```
CREATE TABLE `account` (
  `code` int unsigned NOT NULL AUTO_INCREMENT,
  `user_id` int unsigned NOT NULL,
  `balance` decimal(15,2) unsigned NOT NULL
DEFAULT '0.00',
  PRIMARY KEY (`code`),
  CONSTRAINT `account_user_id_fk`
   FOREIGN KEY (`user_id`)
   REFERENCES `user` (`id`)
    ON UPDATE CASCADE
    ON DELETE NO ACTION,
  CONSTRAINT `account_balance_ck`
   CHECK (`balance` >= 0)
) ENGINE=InnoDB
```

Database schema (2/2)

```
CREATE TABLE `transfer` (
  `id`                       int unsigned NOT NULL AUTO_INCREMENT,
  `timestamp`                timestamp NOT NULL DEFAULT current_timestamp,
  `account_code_orderer`     int unsigned NOT NULL,
  `account_code_beneficiary` int unsigned NOT NULL,
  `amount` decimal(15,2)     unsigned NOT NULL,
  `reason` varchar(255)      NOT NULL,
  PRIMARY KEY (`id`),
  CONSTRAINT `transfer_account_code_beneficiary_fk` FOREIGN KEY (`account_code_beneficiary`)
   REFERENCES `account` (`code`) ON UPDATE CASCADE ON DELETE NO ACTION,
  CONSTRAINT `transfer_account_code_orderer_fk` FOREIGN KEY (`account_code_orderer`)
   REFERENCES `account` (`code`) ON UPDATE CASCADE ON DELETE NO ACTION,
  CONSTRAINT `transfer_amount_ck` CHECK (`amount` > 0)
) ENGINE=InnoDB
```

Database content (1/2)

user

id	name	surname	email	username	password
1	Giovanni	Manfredi	giovanni@polimi.it	Gio	passwordDifficile23
2	Sebastiano	Meneghin	sebastiano@polimi.it	Seba	passwordDifficile25
3	Jeff	Bezos	boss@amazon.com	Jeffrey	youCanDoIt
4	Jim	Gray	admin@ibm.com	Admin	admin

account

code	user_id	balance
1	1	1000.05
2	2	2000.99
3	3	1000000.85
4	3	2000000.95
5	1	285.60
6	4	4000

Database content (2/2)

transfer

id	timestamp	code_account_orderer	code_account_beneficiary	amount	reason
1	2022-03-03 00:05:55	3	1	1000.00	project funding
2	2022-04-05 10:35:01	5	2	500.00	cash
3	2022-08-05 03:35:01	3	6	200.00	refund

Analisi requisiti applicazione

Un'applicazione web consente la gestione di trasferimenti di denaro online da un conto a un altro. 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 utente ha un nome, un cognome, uno username e uno o più conti correnti. Un conto ha un codice, un saldo, e i trasferimenti fatti (in uscita) e ricevuti (in ingresso) dal conto. Un trasferimento ha una data, un importo, un conto di origine e un conto di destinazione. Quando l'utente accede all'applicazione appare una pagina LOGIN per la verifica delle credenziali. In seguito all'autenticazione dell'utente appare l'HOME page che mostra l'elenco dei suoi conti. Quando l'utente seleziona un conto, appare una pagina STATO DEL CONTO che mostra i dettagli del conto e la lista dei movimenti in entrata e in uscita, ordinati per data discendente. La pagina contiene anche una form per ordinare un trasferimento. La form contiene i campi: codice utente destinatario, codice conto destinatario, causale e importo. All'invio della form con il bottone INVIA, l'applicazione controlla che il conto di destinazione appartenga all'utente specificato e che il conto origine abbia un saldo superiore o uguale all'importo del trasferimento. In caso di mancanza di anche solo una condizione, l'applicazione mostra una pagina con un avviso di fallimento che spiega il motivo del mancato trasferimento. Nel caso in cui entrambe le condizioni siano soddisfatte, l'applicazione deduce l'importo dal conto di origine, aggiunge l'importo al conto di destinazione e mostra una pagina CONFERMA TRASFERIMENTO che presenta i dati dell'importo trasferito e i dati del conto di origine e di destinazione con i rispettivi saldi precedenti al trasferimento e aggiornati dopo il trasferimento. L'applicazione deve garantire l'atomicità del trasferimento: ogni volta che il conto di destinazione viene addebitato, il conto di origine deve essere accreditato. Ogni pagina contiene un collegamento per tornare alla pagina precedente. L'applicazione consente il logout dell'utente.

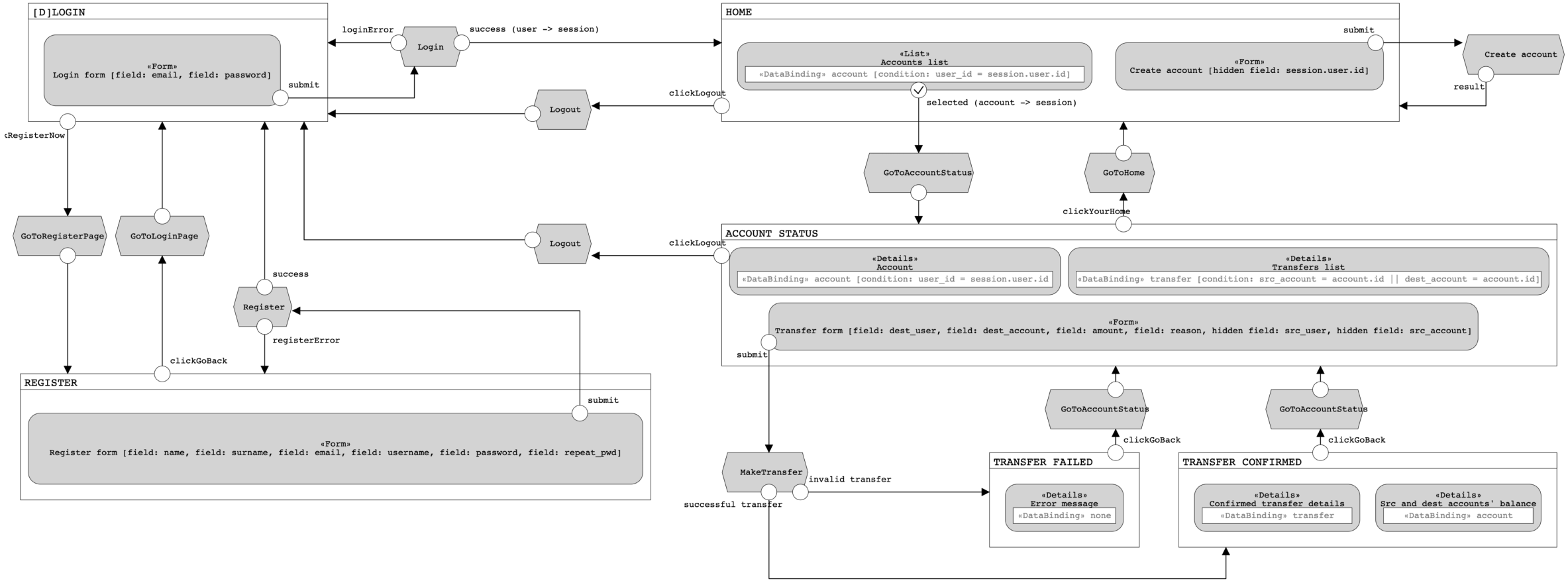
Pagine, componenti, eventi, azioni

Completamento delle specifiche (2/2) - applicazione

- Nelle pagine di LOGIN e REGISTER sono presenti dei bottoni per passare rispettivamente da una pagina all'altra
- Nella HOME ad ogni conto (account) in elenco viene affiancato il saldo (balance) del conto
- Per credenziali di login si intendono email e password
- Se un utente prova ad accedere alle pagine di LOGIN e REGISTER ed è ancora loggato, verrà reindirizzato automaticamente alla HOME
- Se si verificano azioni impreviste (accesso ad aree protette del sito senza averne l'autorizzazione oppure attributi di richiesta invalidi), l'utente viene reindirizzato ad una pagina di ERRORE contentente il motivo dell'errore e con un link alla LOGIN page
- Nella pagina HOME è anche possibile per l'utente creare un nuovo conto
- Se un nuovo utente si registra verrà creato automaticamente un conto a lui collegato con saldo a zero. Non è quindi possibile che un utente non abbia un account a lui collegato
- Dopo la registrazione l'utente deve passare dalla pagina di LOGIN

Pagine, componenti, eventi, azioni

Design applicativo (IFML)



Riferirsi ai sequence diagrams per lo sviluppo dettagliato delle interazioni nell'applicazione

Components

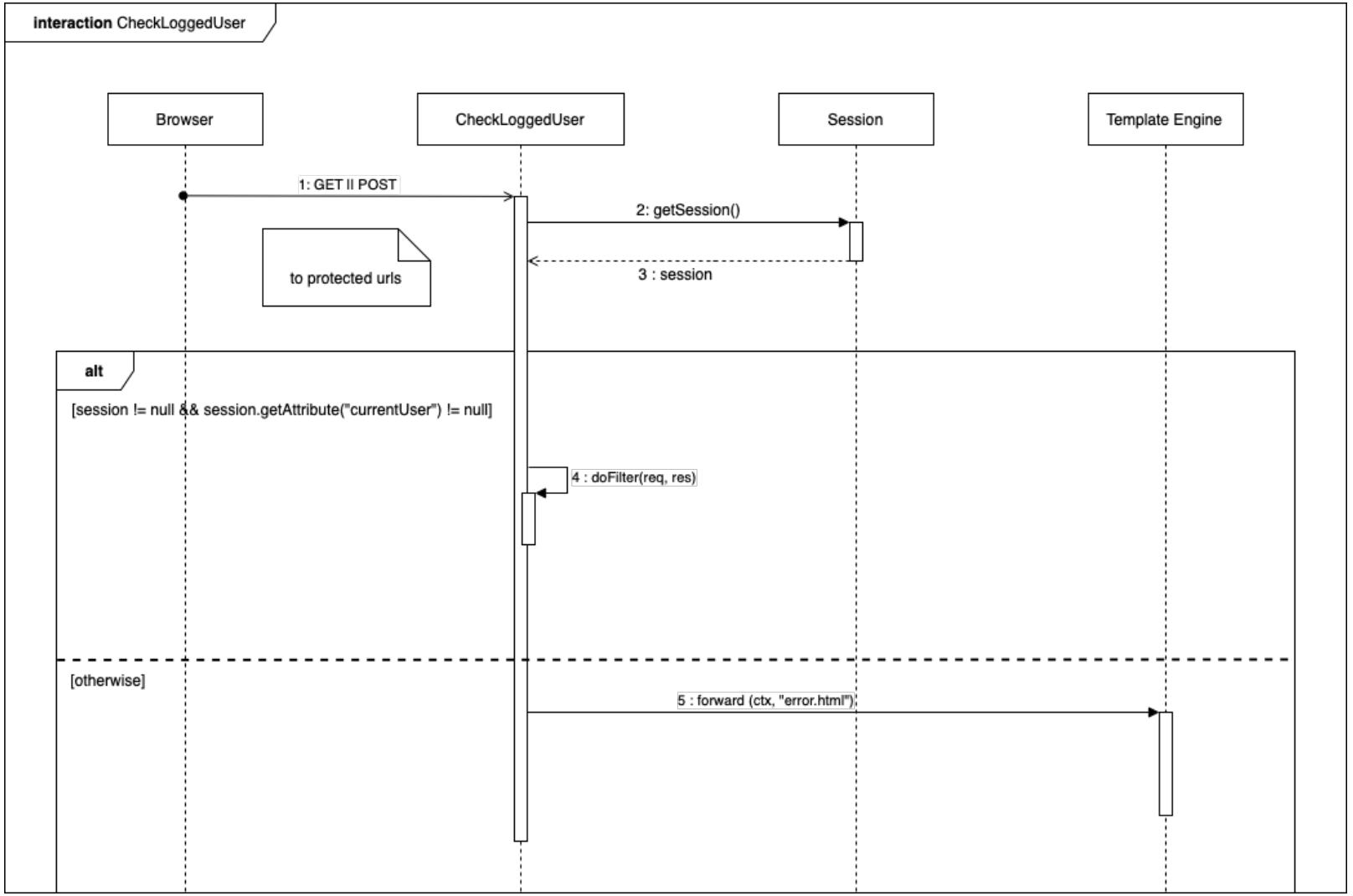
- Model objects (beans)
  - User
  - Account
  - Transfer
- Data Access Objects (classes)
  - UserDAO
    - findUser(email, password) : User
    - findUserById(id) : User
    - findUserByEmail(email) : User
    - findUserByUsername(username) : User
    - createUser(name, surname, email, username, password) : void
    - registerUser (name, surname, email, username, password) : void
    - registerUser (name, surname, email, username, password, balance) : void
  - AccountDAO
    - findAccountByCode(code) : Account
    - findAccountsByUserId(user_id) : List<Account>
    - createAccount(user_id) : void
    - createAccount(user_id, balance) : void
  - TransferDAO
    - findTransfersByAccountCode(code) : List<Transfer>
    - createTransfer(code_account_orderer, code_account_beneficiary, amount, reason) : void
- Filters
  - CheckLoggedUser
  - CheckNotLoggedUser
- Controllers (serlvets)
  - Login
  - Logout
  - Register
  - GoToLoginPage
  - GoToRegisterPage
  - GoToHome
  - CreateAccount
  - GoToAccountStatus
  - MakeTransfer
  - GoToTransferConfirmedPage
- Views (templates)
  - login.html
  - register.html
  - infoPane.html
  - error.html
  - home.html
  - accountStatus.html
  - transferFailed.html
  - transferConfirmed.html

Sequence diagrams

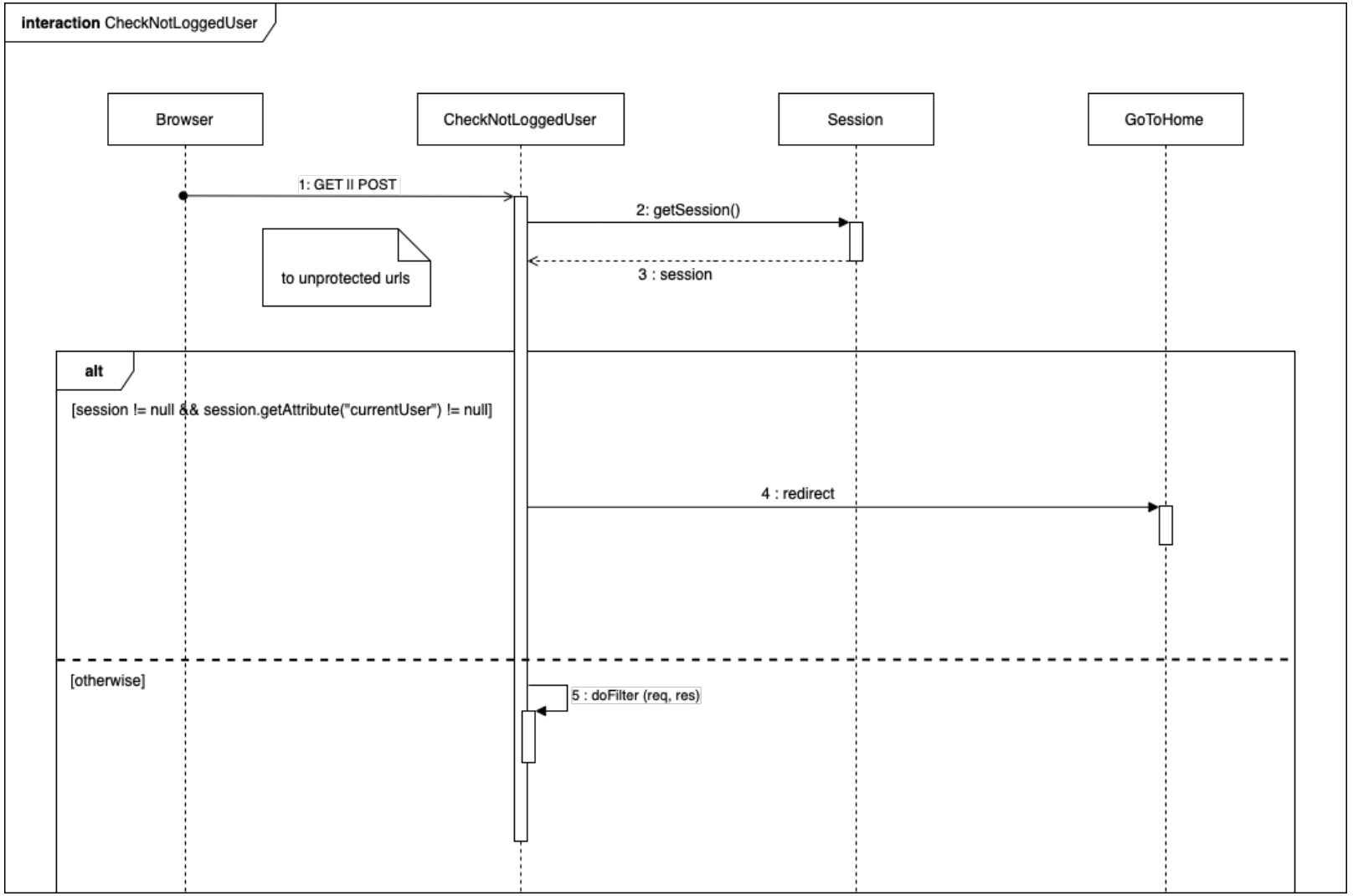
I diagrammi di sequenza seguenti mirano a illustrare l'interazione degli eventi principali dell'applicazione web. Si è cercato di utilizzare il maggior grado di dettaglio possibile nella rappresentazione ma, a favore di una migliore chiarezza e minore ripetitività, alcuni elementi sono stati omessi dopo la loro prima rappresentazione (es. errori interni al server, errori di accesso interni al database, parametri *null*).

Inoltre i controlli effettuati sui filtri rappresentati nei primi diagrammi sono stati omessi nelle chiamate successive, che vengono però svolte correttamente. Per un maggior dettaglio sulle chiamate dei filtri si consiglia di visionare il file `web.xml`.

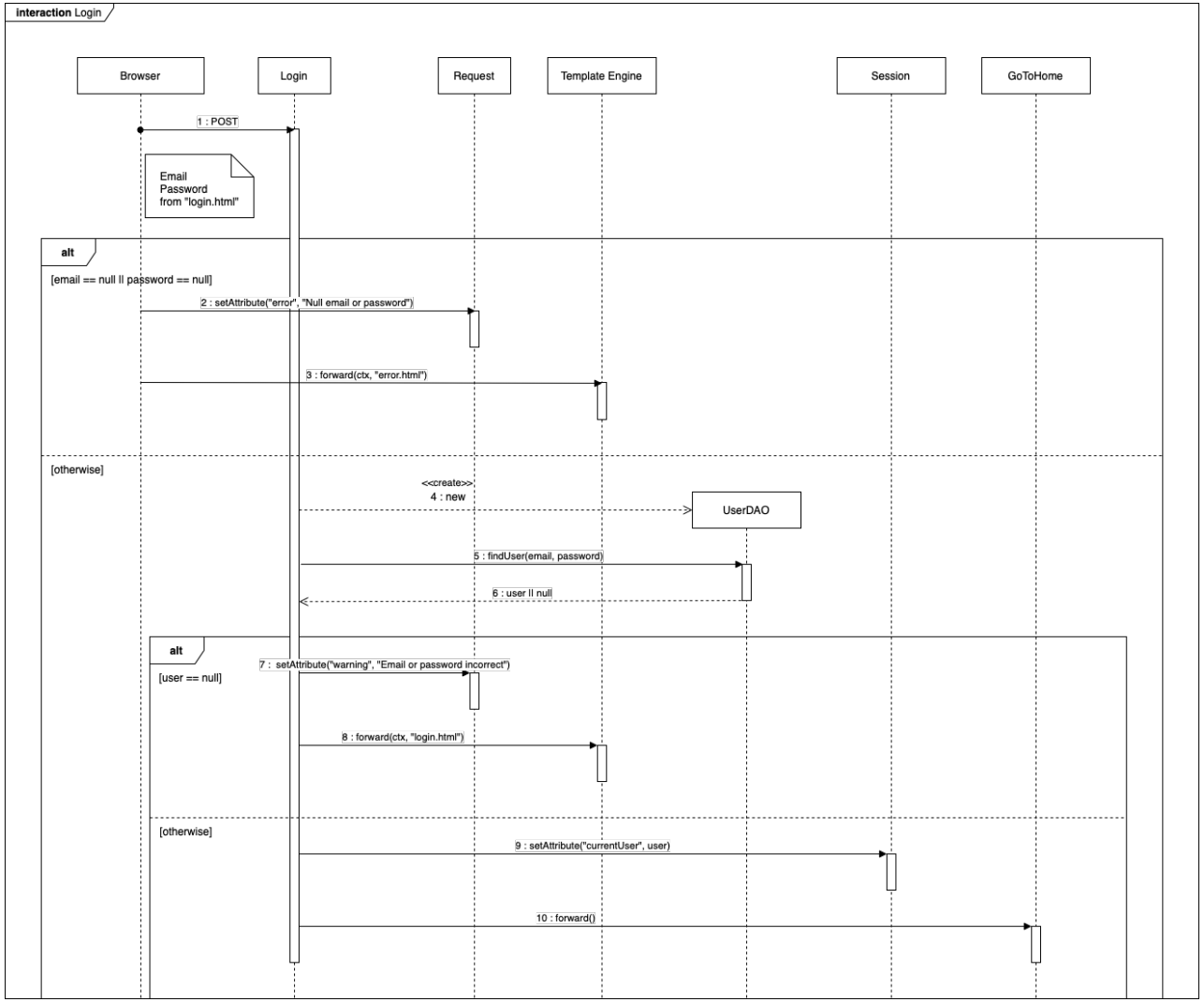
CheckLoggedUser



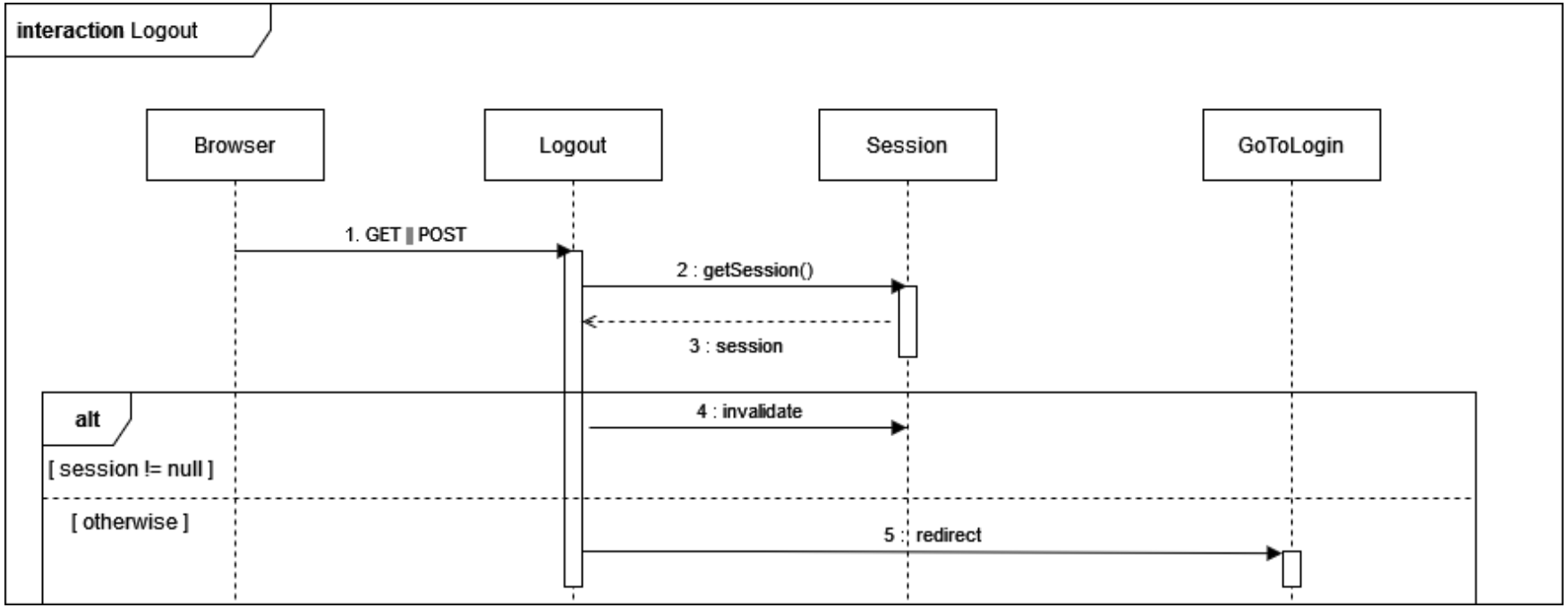
CheckNotLoggedUser



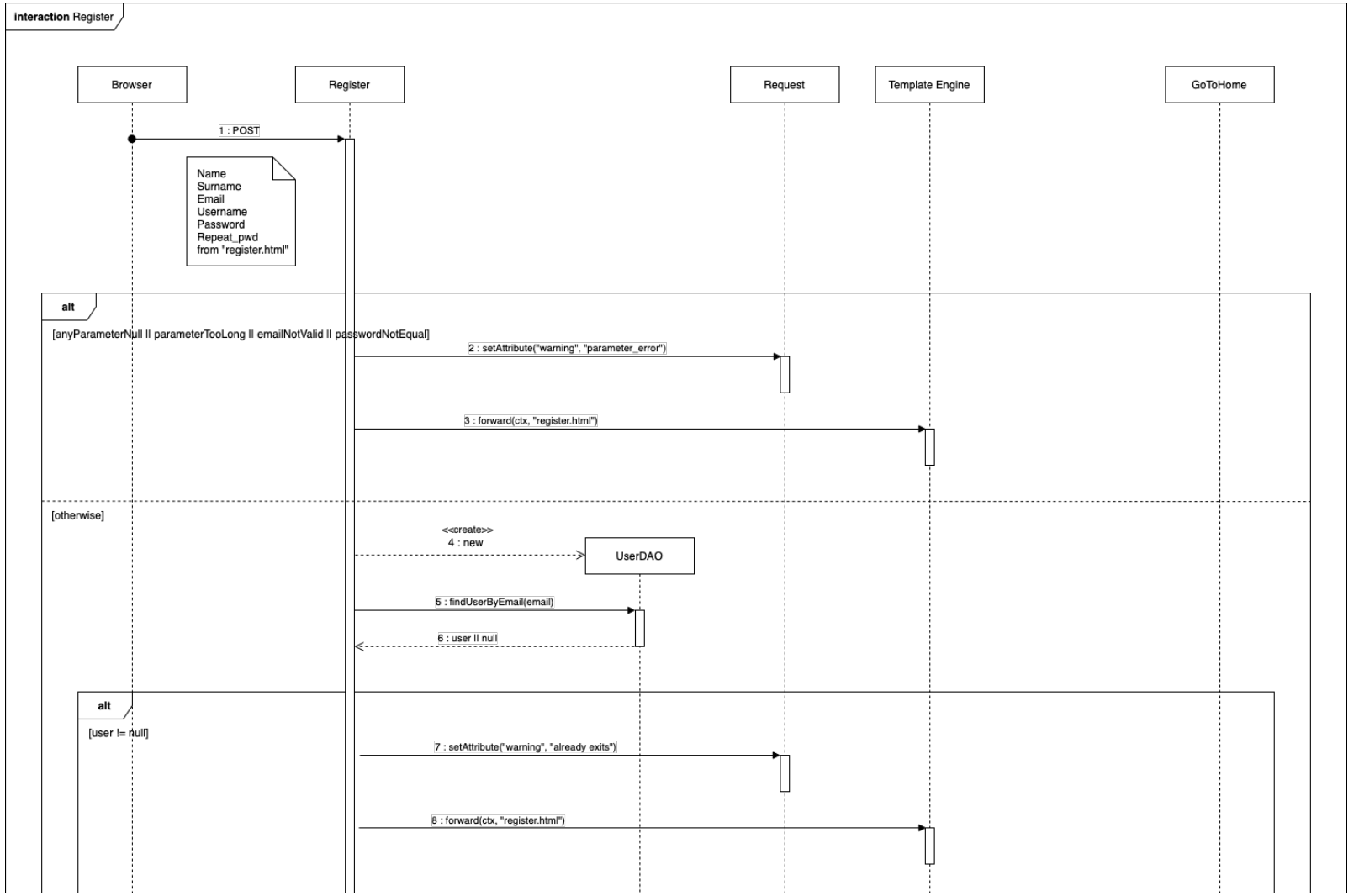
Login


interaction Login
Browser
Login
Request
Template Engine
Session
GoToHome
1 : POST
Email
Password
from "login.html"
alt
[email == null || password == null]
2 : setAttribute("error", "Null email or password")
3 : forward(ctx, "error.html")
[otherwise]
<<create>>
4 : new
UserDAO
5 : findUser(email, password)
6 : user || null
alt
[user == null]
7 : setAttribute("warning", "Email or password incorrect")
8 : forward(ctx, "login.html")
[otherwise]
9 : setAttribute("currentUser", user)
10 : forward()

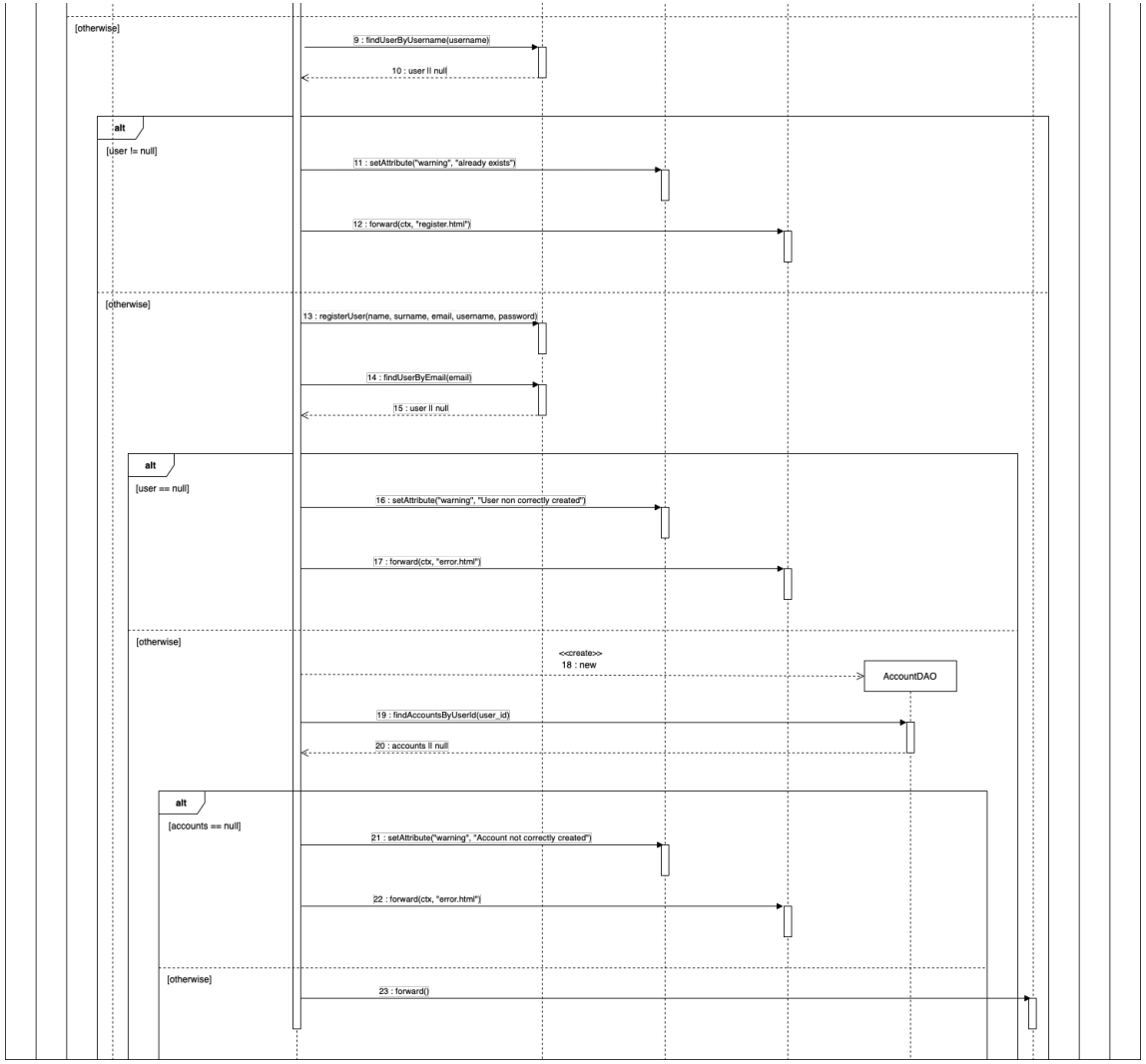
Logout


interaction Logout
Browser
Logout
Session
GoToLogin
1. GET || POST
2 : getSession()
3 : session
alt
[session != null]
4 : invalidate
[otherwise]
5 : redirect

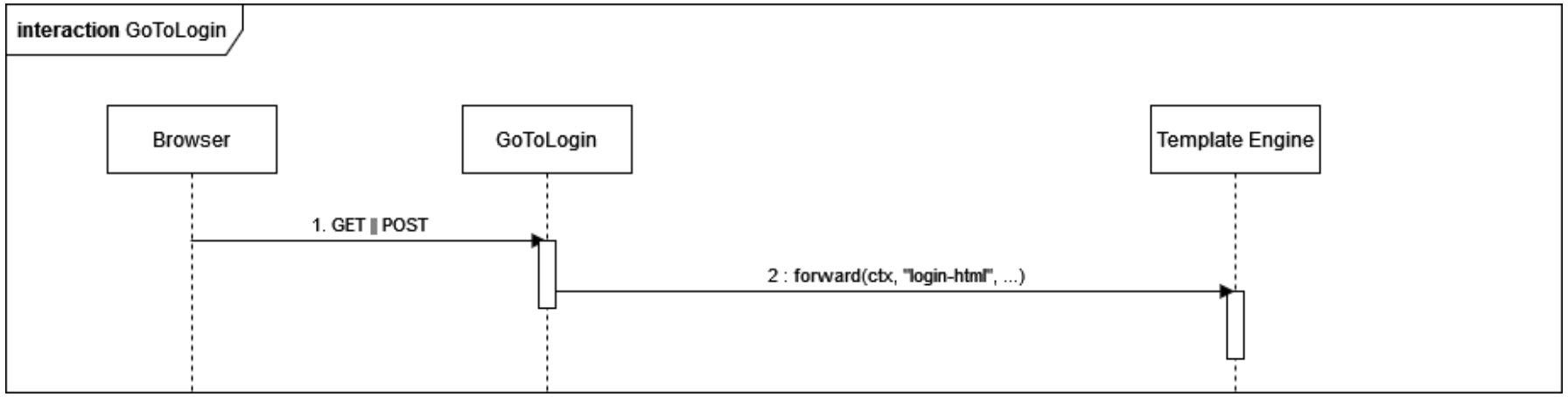
Register (1/2)


interaction Register
Browser
Register
Request
Template Engine
GoToHome
1 : POST
Name
Surname
Email
Username
Password
Repeat_pwd
from "register.html"
alt
[anyParameterNull || parameterTooLong || emailNotValid || passwordNotEqual]
2 : setAttribute("warning", "parameter_error")
3 : forward(ctx, "register.html")
[otherwise]
<<create>>
4 : new
UserDAO
5 : findUserByEmail(email)
6 : user || null
alt
[user != null]
7 : setAttribute("warning", "already exits")
8 : forward(ctx, "register.html")

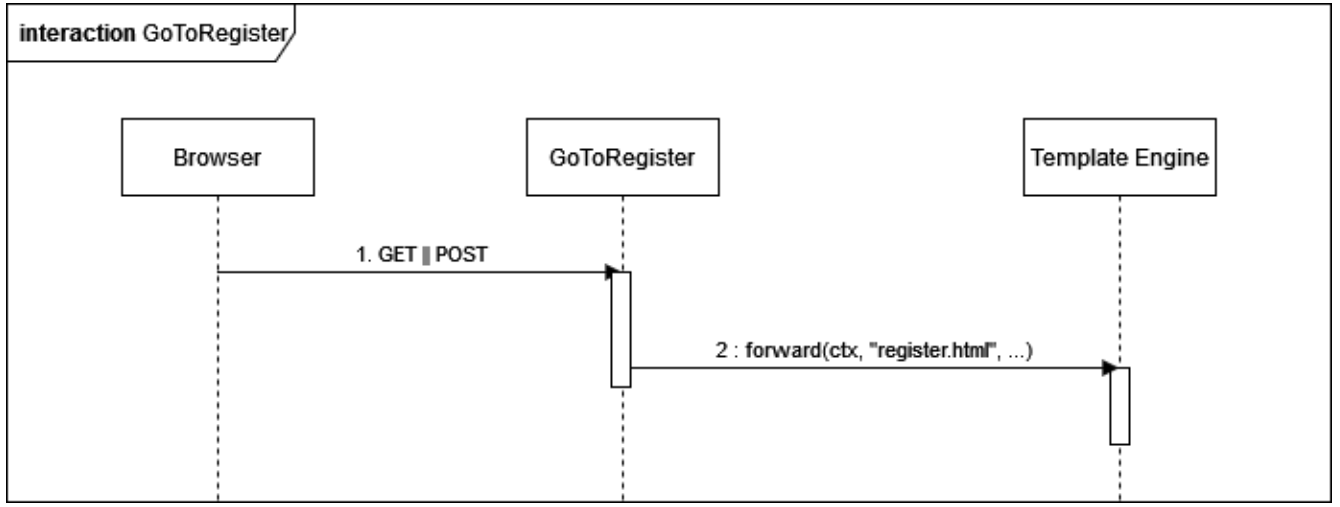
Register (2/2)


[otherwise]
9 : findUserByUsername(username)
10 : user || null
alt
[user != null]
11 : setAttribute("warning", "already exists")
12 : forward(ctx, "register.html")
[otherwise]
13 : registerUser(name, surname, email, username, password)
14 : findUserByEmail(email)
15 : user || null
alt
[user == null]
16 : setAttribute("warning", "User non correctly created")
17 : forward(ctx, "error.html")
[otherwise]
<<create>>
18 : new
AccountDAO
19 : findAccountsByUserId(user_id)
20 : accounts || null
alt
[accounts == null]
21 : setAttribute("warning", "Account not correctly created")
22 : forward(ctx, "error.html")
[otherwise]
23 : forward()

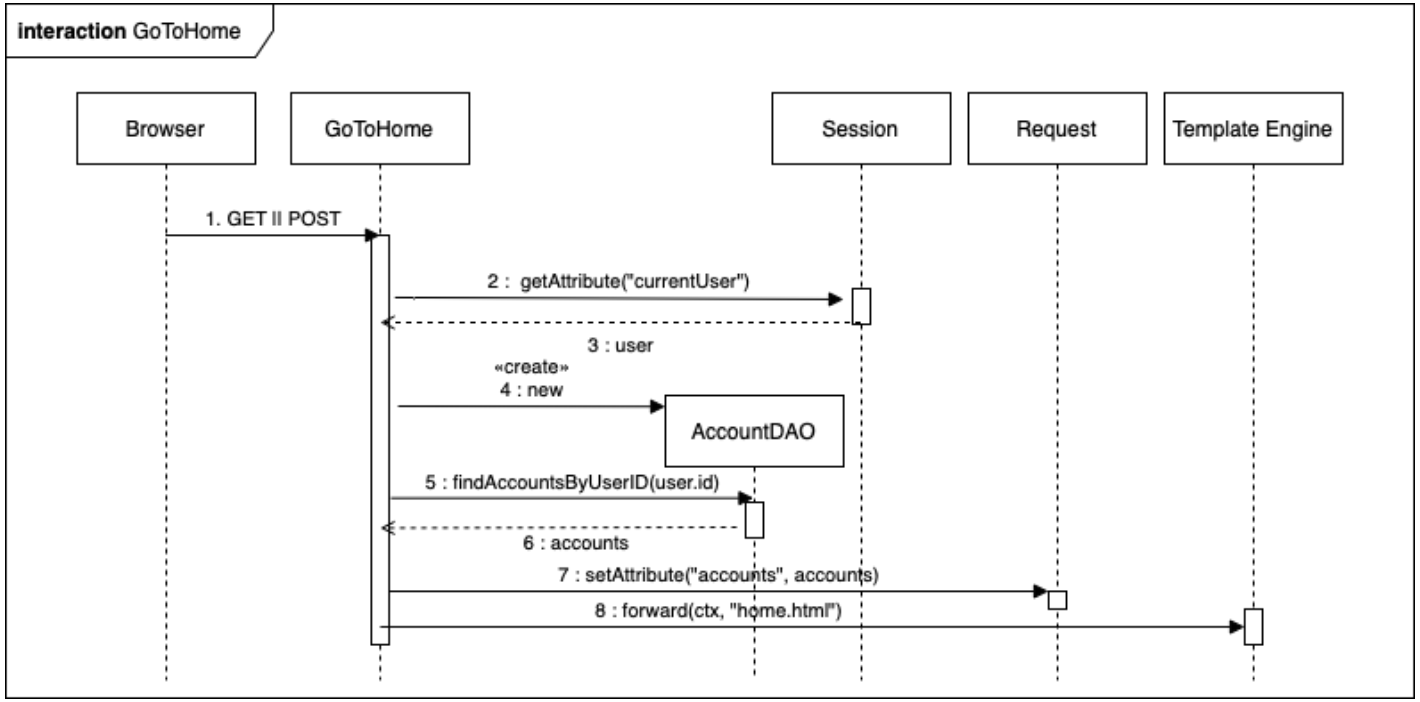
GoToLogin


interaction GoToLogin
Browser
GoToLogin
Template Engine
1. GET || POST
2 : forward(ctx, "login-html", ...)

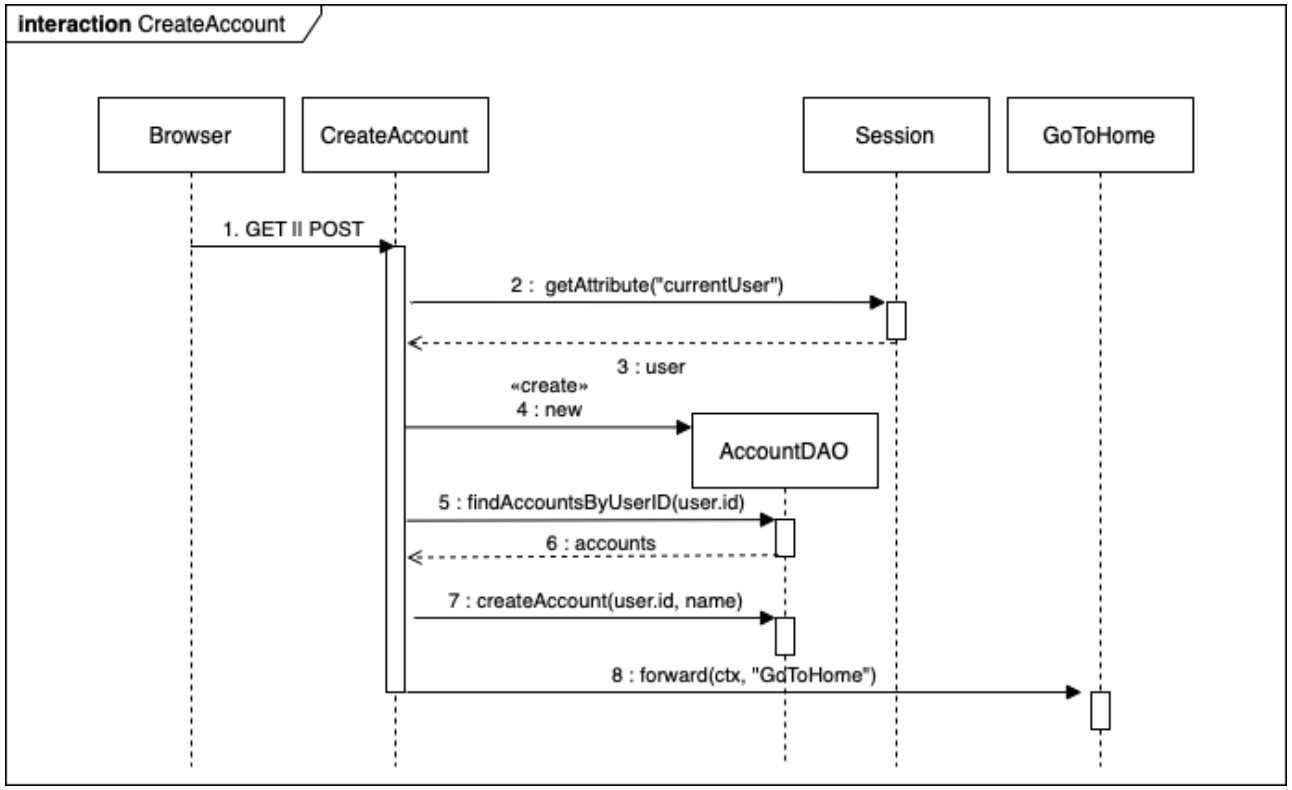
GoToRegister



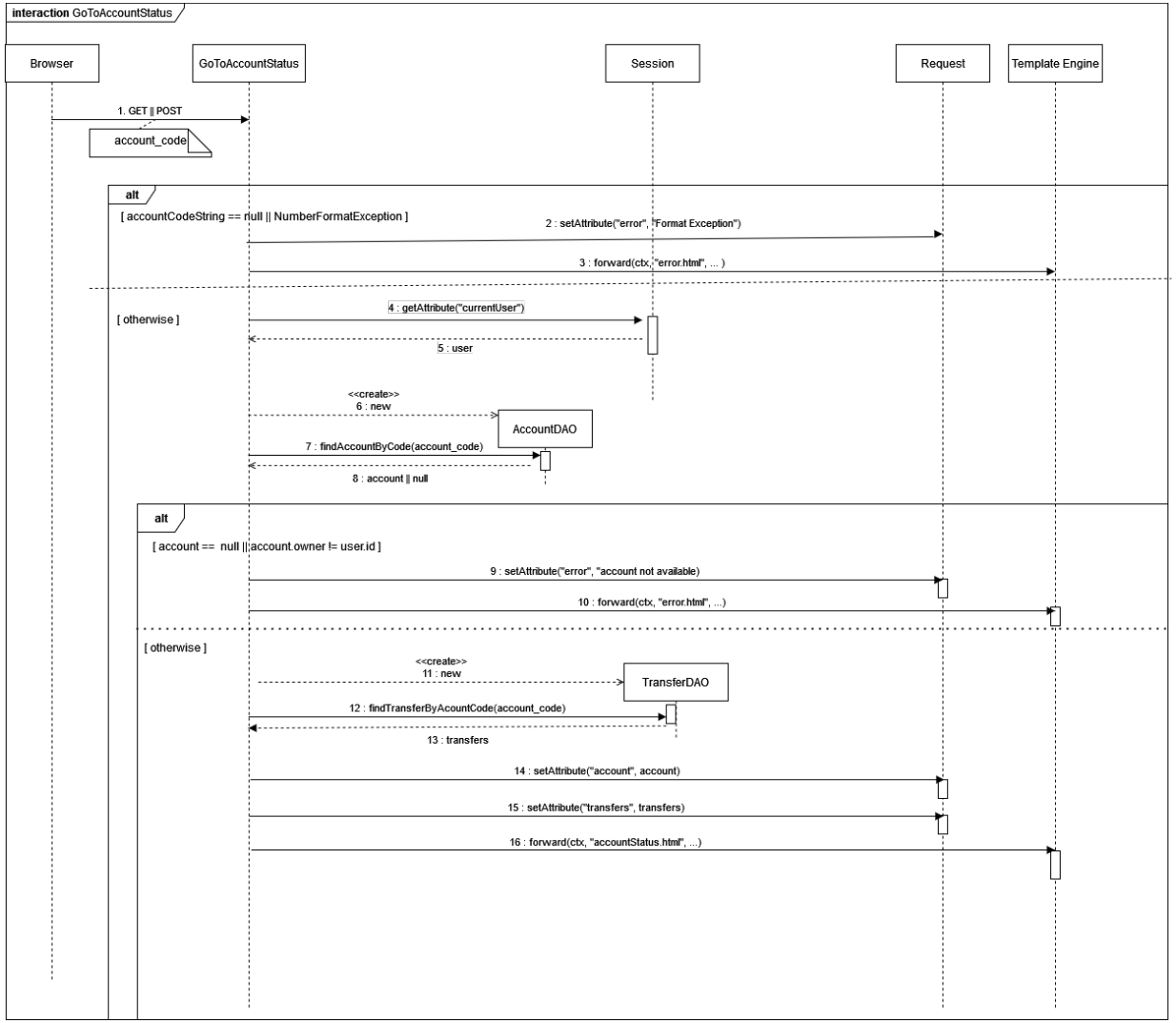
GoToHome


interaction GoToHome
Browser
GoToHome
Session
Request
Template Engine
1. GET II POST
2 : getAttribute("currentUser")
3 : user
«create»
4 : new
AccountDAO
5 : findAccountsByUserID(user.id)
6 : accounts
7 : setAttribute("accounts", accounts)
8 : forward(ctx, "home.html")

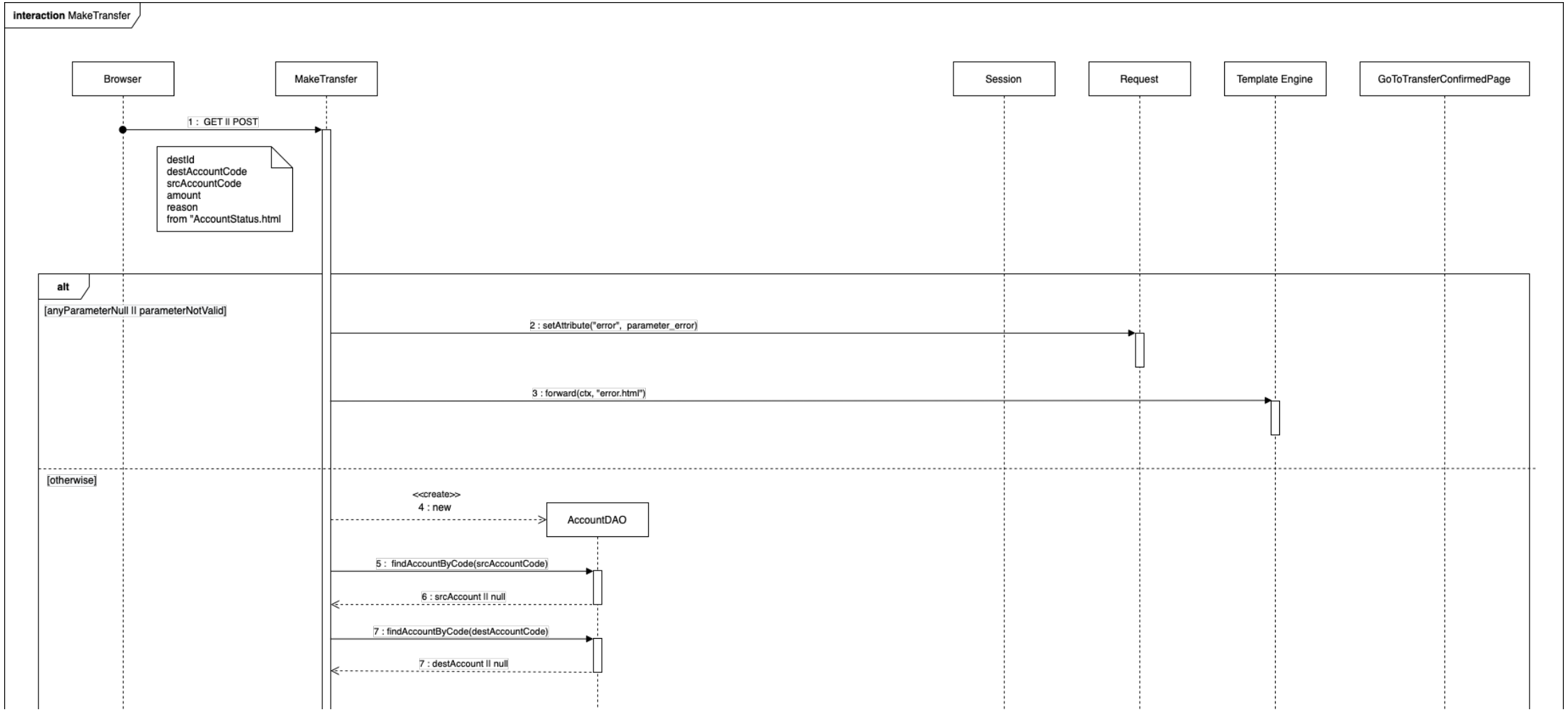
CreateAccount



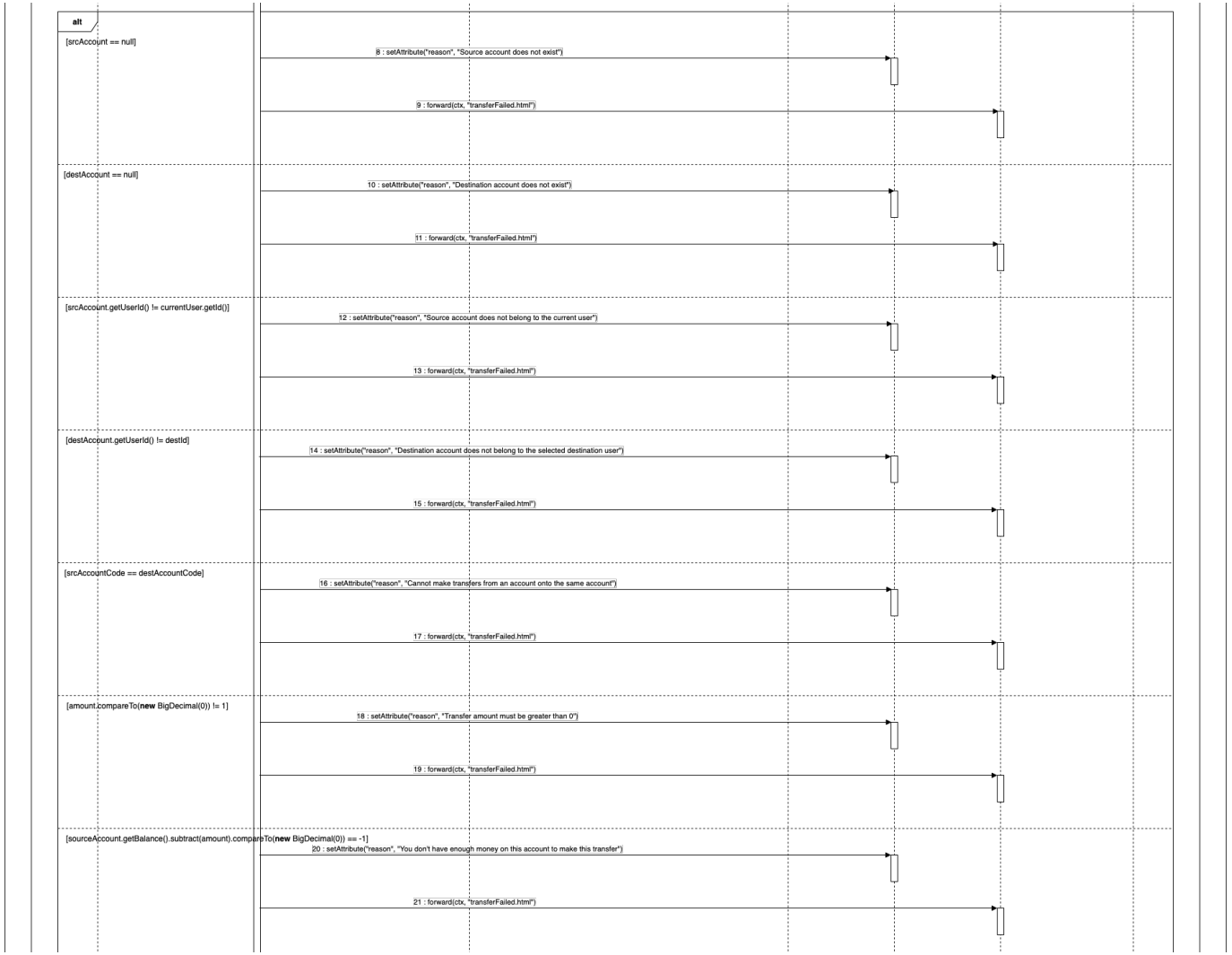
GoToAccountStatus



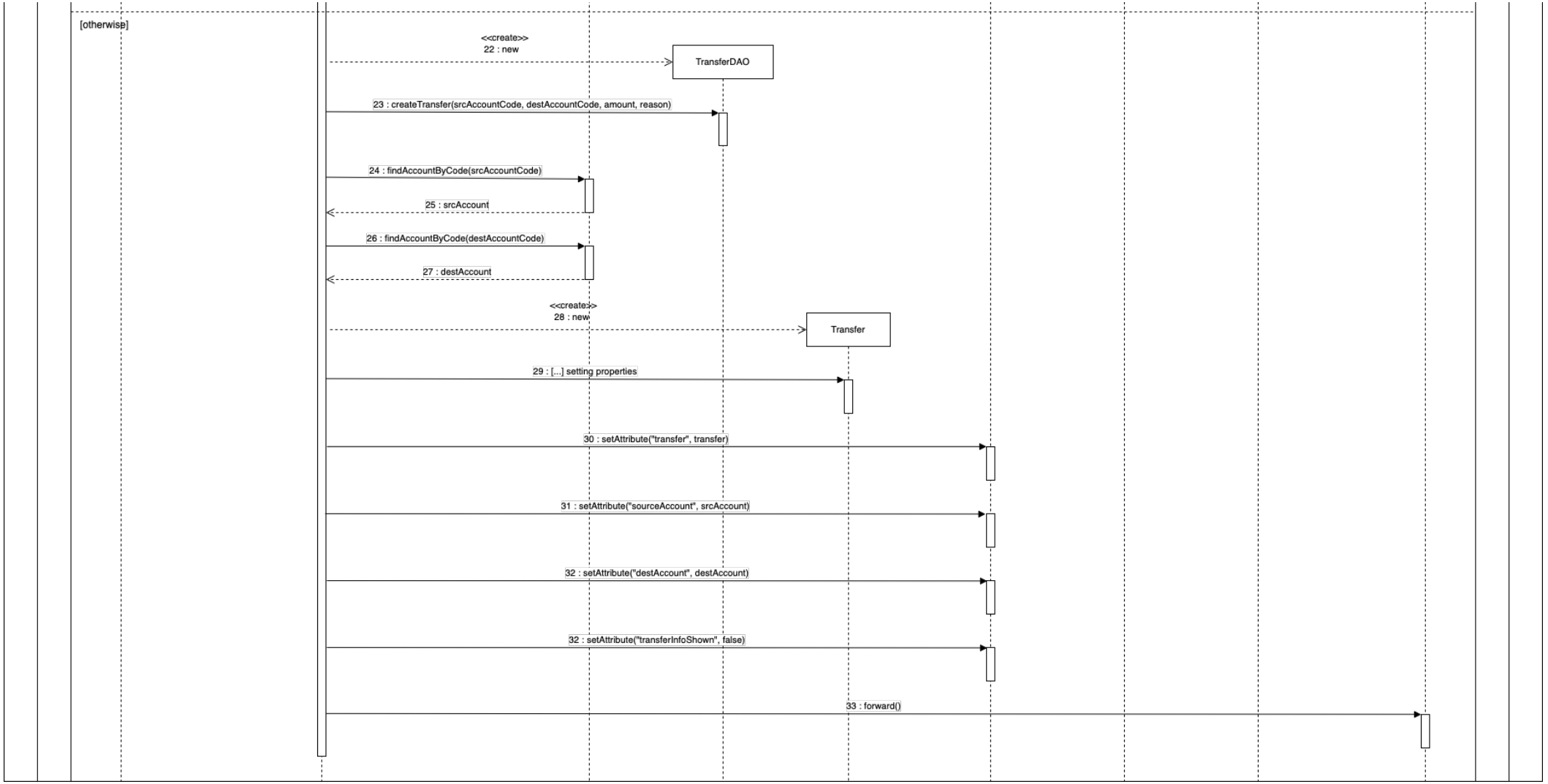
MakeTransfer (1/3)



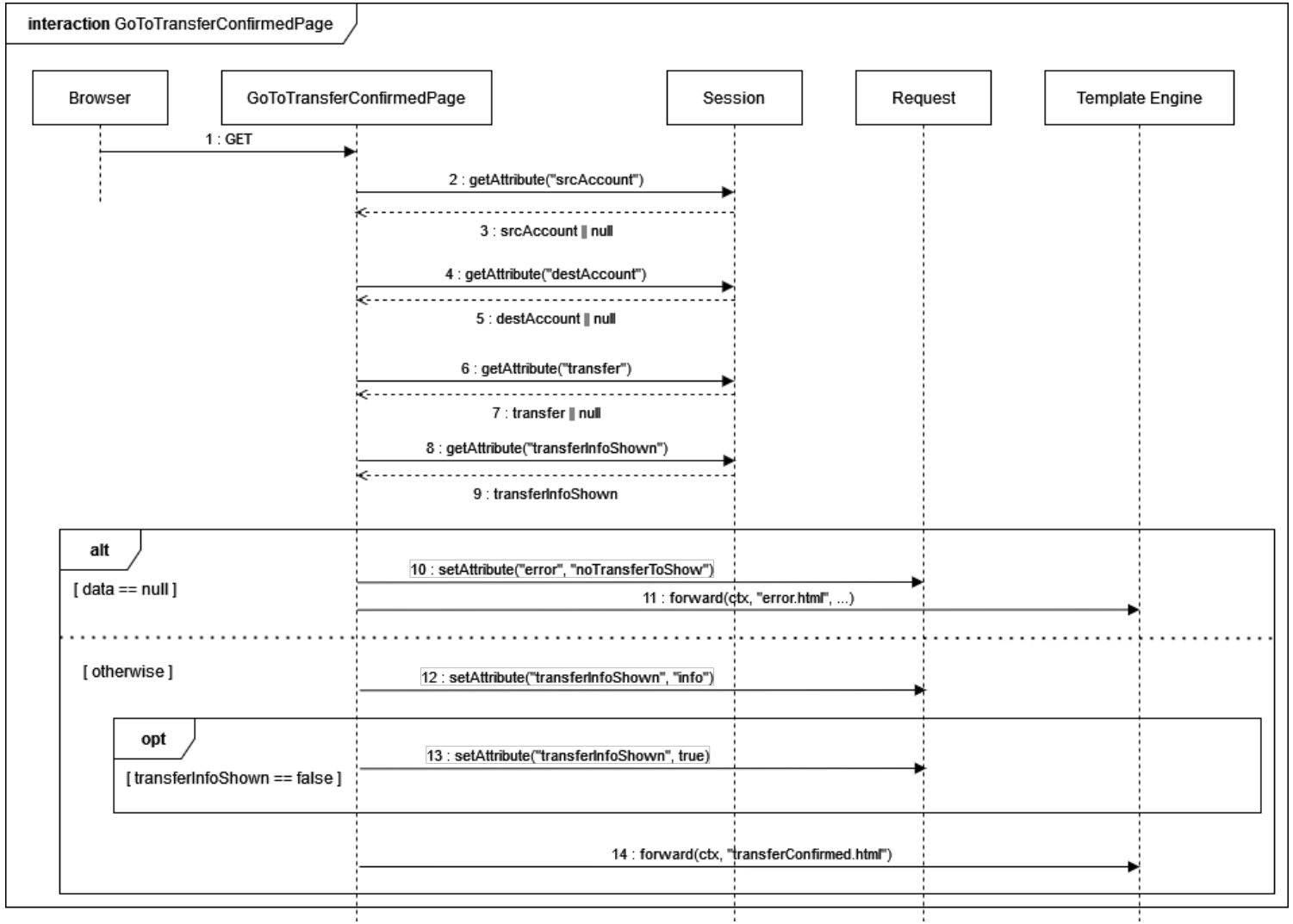
MakeTransfer (2/3)



MakeTransfer (3/3)



GoToTransferConfirmedPage



Alcuni link utili

- DAO, chisura dei preparedStatement e resultSet
  https://stackoverflow.com/questions/14546592/do-i-need-to-close-preparedstatement
- GET Generated Keys, estrarre le chiavi generate da una entry sul DB
  https://stackoverflow.com/questions/4224228/preparedstatement-with-statement-return-generated-keys
- GET parameters in thymeleaf
  https://stackoverflow.com/questions/39865482/how-can-i-call-getters-from-model-passed-to-thymeleaf-like-parameter
- Validating an email address
  https://stackoverflow.com/questions/201323/how-can-i-validate-an-email-address-using-a-regular-expression
  https://regexr.com
- Thymeleaf guide
  https://www.thymeleaf.org/doc/tutorials/3.0/usingthymeleaf.html
- Altri link nel codice del progetto