Tecnologie informatiche per il web

Traccia 4: trasferimento denaro

Arturo Benedetti

Professore: Piero Fraternali

POLITECNICO
MILANO 1863

Analisi dei dati

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** ed 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.

Entities, **attributes**, **relationships**

Analisi dei dati

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 ed 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.

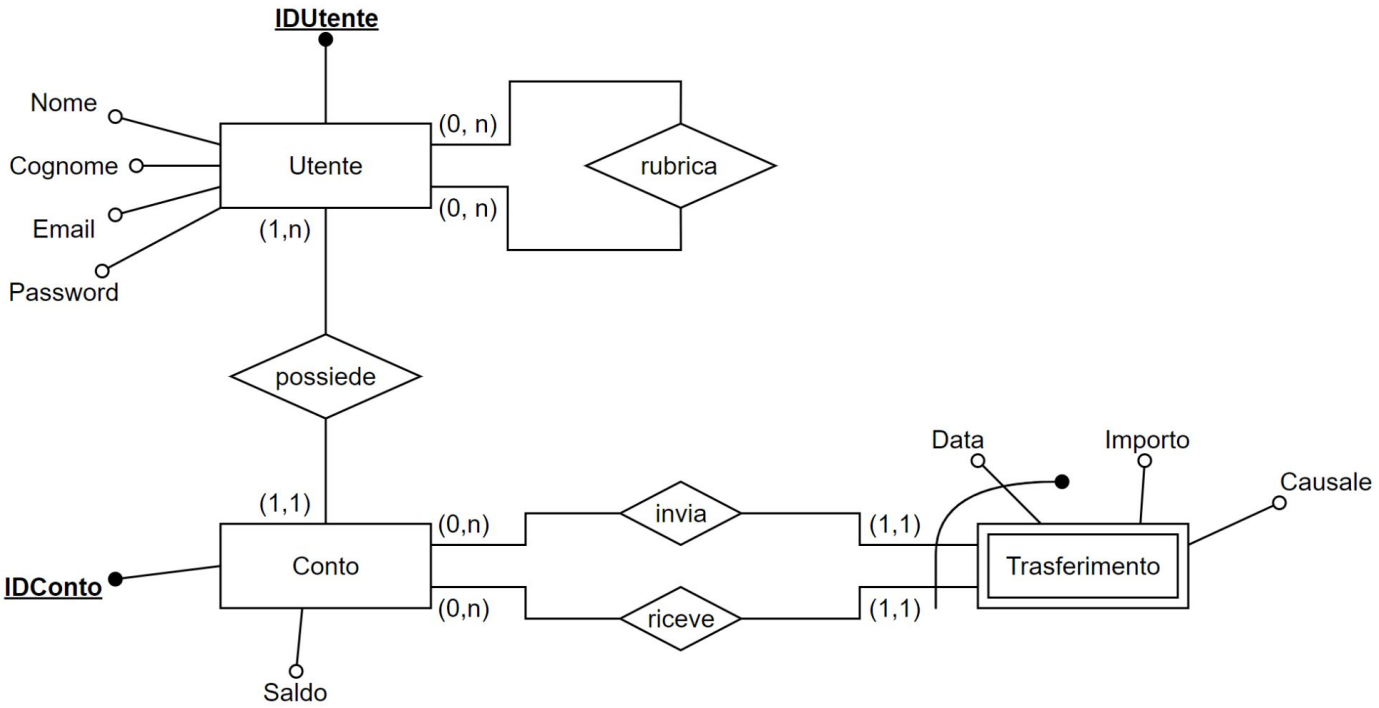
Pages (views), **view components**, **events**, **actions**

Completamento specifica

- La mail dell’utente è considerata essere il suo username.
- La versione con Javascript richiede la funzionalità “salvataggio in rubrica”; in questa versione è quindi utilizzata una relazione aggiuntiva: **rubrica**.
- Tutti i dati di un trasferimento sono obbligatori.
- Non possono essere effettuati trasferimenti con importo negativo o pari a 0.
- Non è possibile effettuare un trasferimento che abbia uguale conto di origine e destinazione.
- La pagina di “conferma trasferimento” è considerata come una pagina per fornire informazioni puntuali al momento del trasferimento, ciò significa che potrebbe non essere “coerente” se tenuta aperta per molto tempo e poi aggiornata.
- Non è possibile effettuare trasferimenti verso conti non esistenti

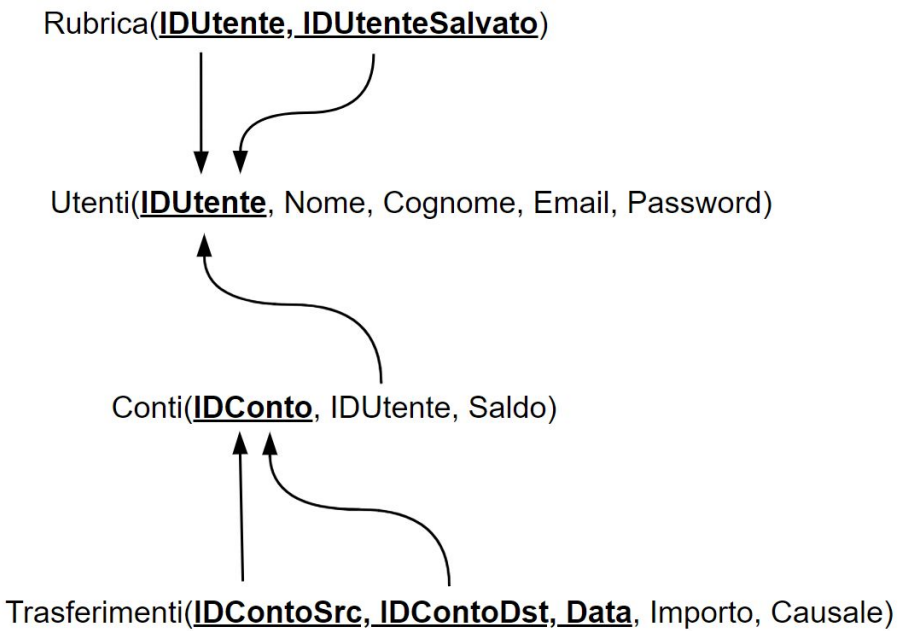
Database design

Concettuale

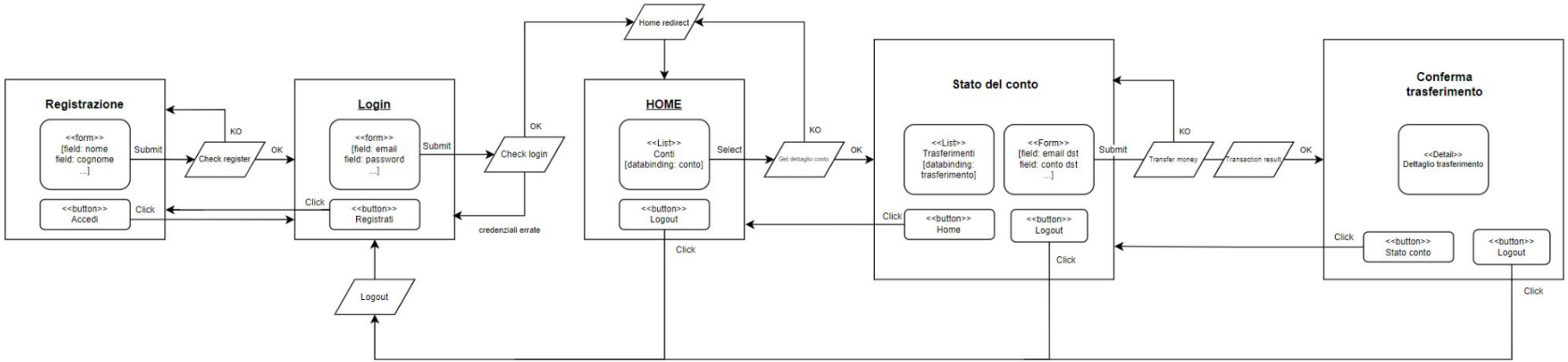


Database design

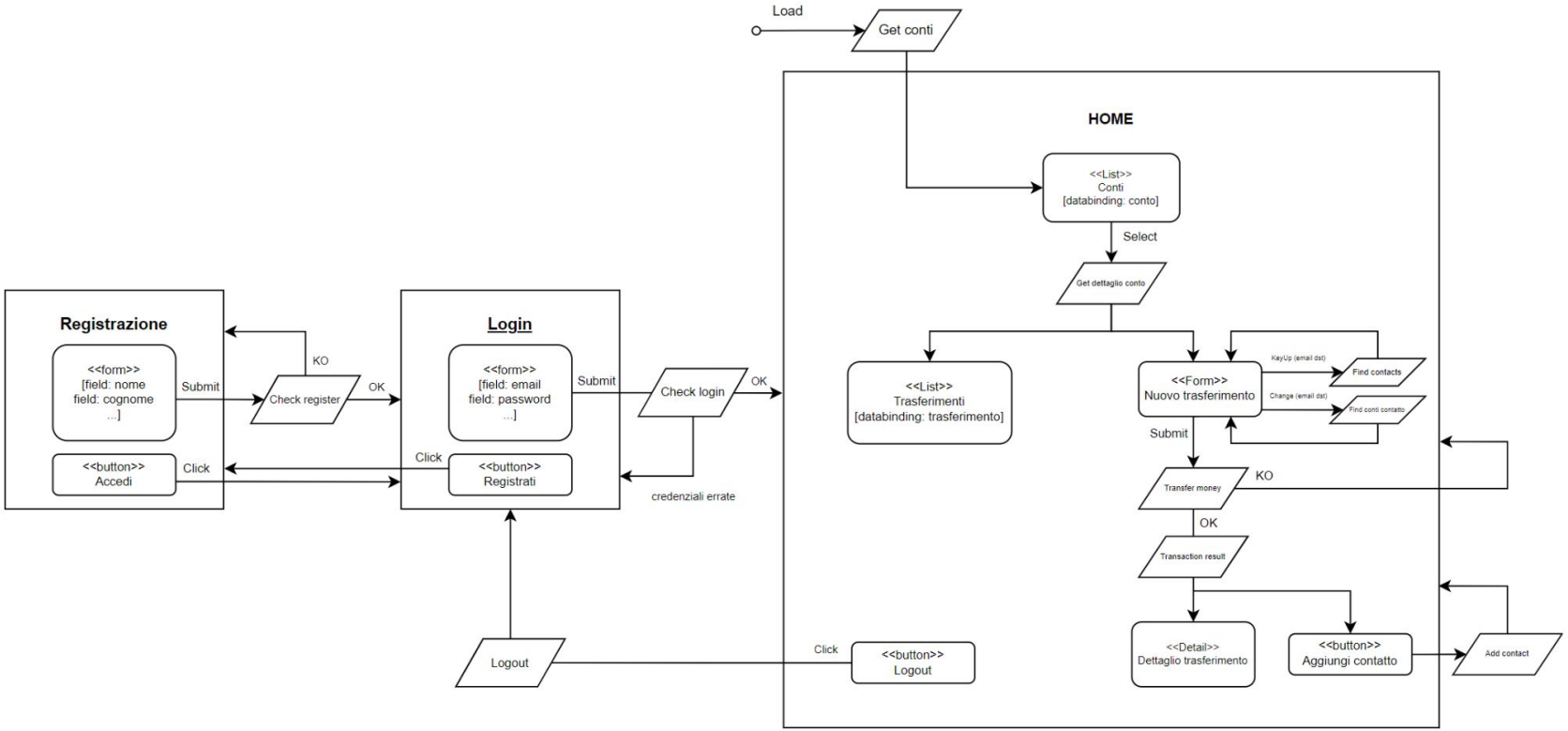
Logico



Application design - HTML



Application design - RIA



Componenti - Server side

- Controllers
  - (AddToContacts)
  - CheckLogin
  - CheckRegister
  - (FindContacts)
  - [HomeRedirect]
  - GetConti
  - (GetContiContatto)
  - GetDettaglioConto
  - Logout
  - TransactionResult
  - TransferMoney

- Model objects (Beans)
  - Utente
  - Conto
  - Trasferimento
  - (Rubrica)

- Data Access Objects (Classes)
  - UtenteDAO
    - checkLogin()
    - checkRegister()
    - registerUser()
    - getUtenteById()
    - getUtenteByEmail()
  - ContoDAO
    - getContiByUtente()
    - (getContiByEmail())
    - getContiById()
  - TrasferimentoDAO
    - transfer()
    - getTrasferimento()
    - getEntrateByConto()
    - getUsciteByConto()
    - getTrasferimentiMail()
    - isLastTrasferimento()
  - (RubricaDAO)
    - addUser()
    - isPresent()
    - getContactByUsername()

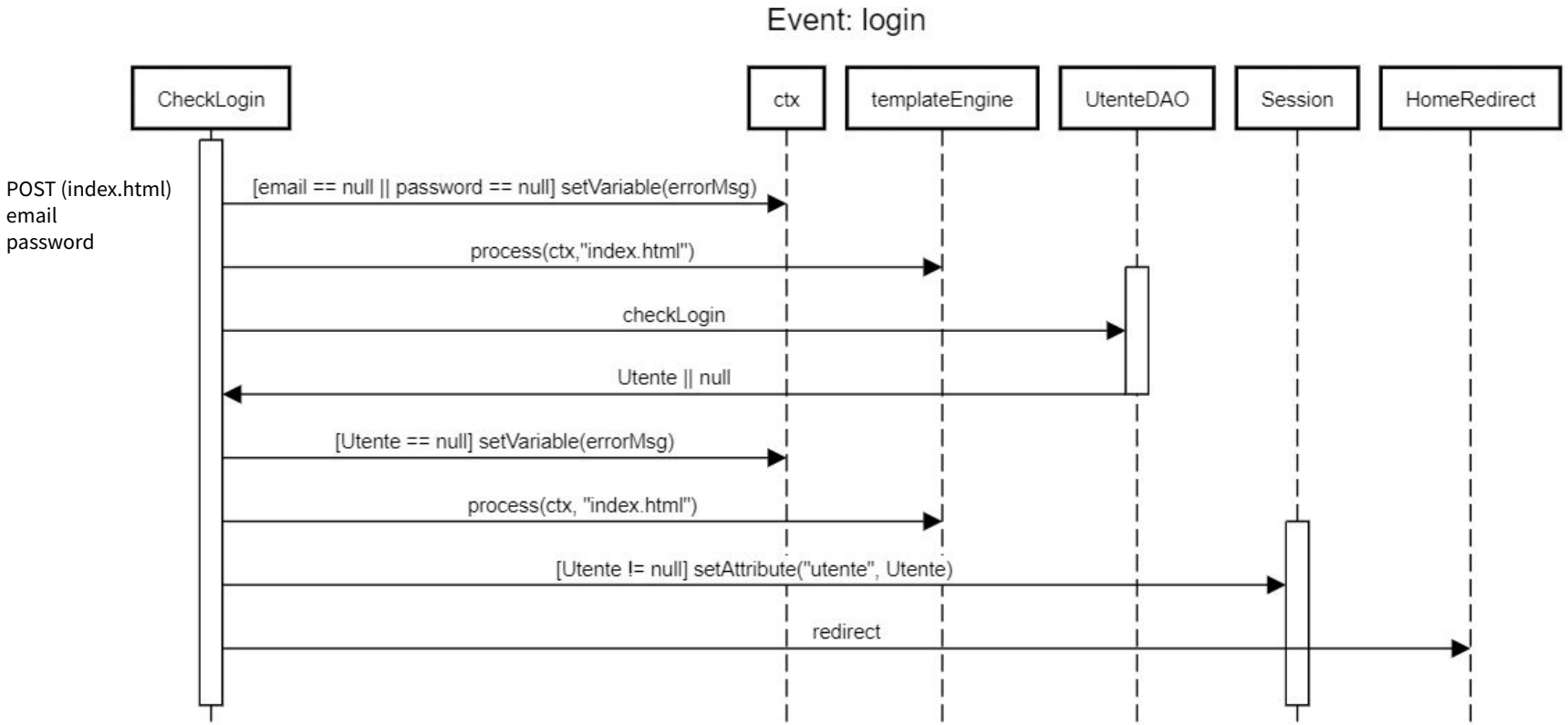
(): Solo RIA
[]: Solo HTML

Componenti - Client side (RIA)

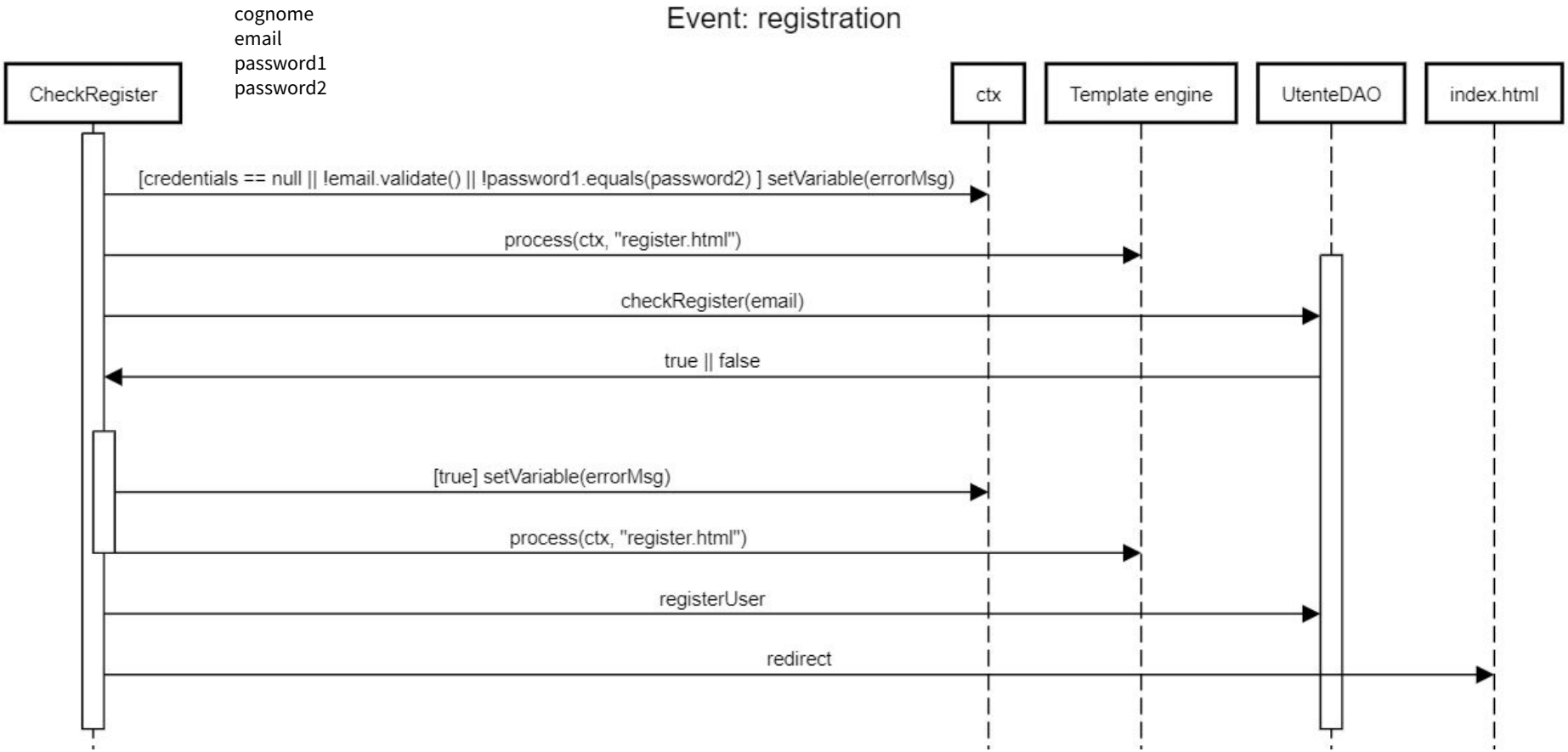
- Index
  - Login form
    - Gestione del submit e degli errori
- Register
  - Registration form
    - Gestione del submit e degli errori
- Home
  - AccountList
    - show(): richiede al server l'elenco dei conti
    - update(): riceve dati dal server ed aggiorna la lista
    - reset(): imposta le condizioni di visibilità iniziali
  - AccountDetails
    - show(): richiede al server l'elenco delle transazioni relative al conto selezionato
    - update(): riceve dati dal server ed aggiorna la lista
    - createTransferButton(): crea ed inizializza un bottone per eseguire un nuovo trasferimento
    - reset(): imposta le condizioni di visibilità iniziali
  - TransactionModal
    - init(): aggiunge i listener necessari per il funzionamento del modal transazioni
    - update(): imposta il riferimento al conto selezionato all'interno di un campo nascosto del modal
    - showMatches(): aggiorna la lista dei contatti da mostrare
    - showContactAccounts(): aggiorna la lista dei conti del destinatario selezionato, o imposta l'unico presente come come campo nascosto
    - reset(): imposta le condizioni di visibilità iniziali

- Funzioni di supporto
  - findMatches(text): richiede al server gli utenti la cui email inizia per il testo passato
  - findAccounts(email): richiede al server i numeri di conto per l'utente specificato
  - getTransactionResult(): richiede al server il risultato di una transazione
  - addContact(): richiede al server il salvataggio in rubrica dell'utente selezionato
  - openModal(modalID): rende visibile il modal
  - closeModal(modalID): nasconde il modal
  - prepareModalTransferimento(): imposta i componenti che compongono il modal per la visualizzazione del risultato della transazione

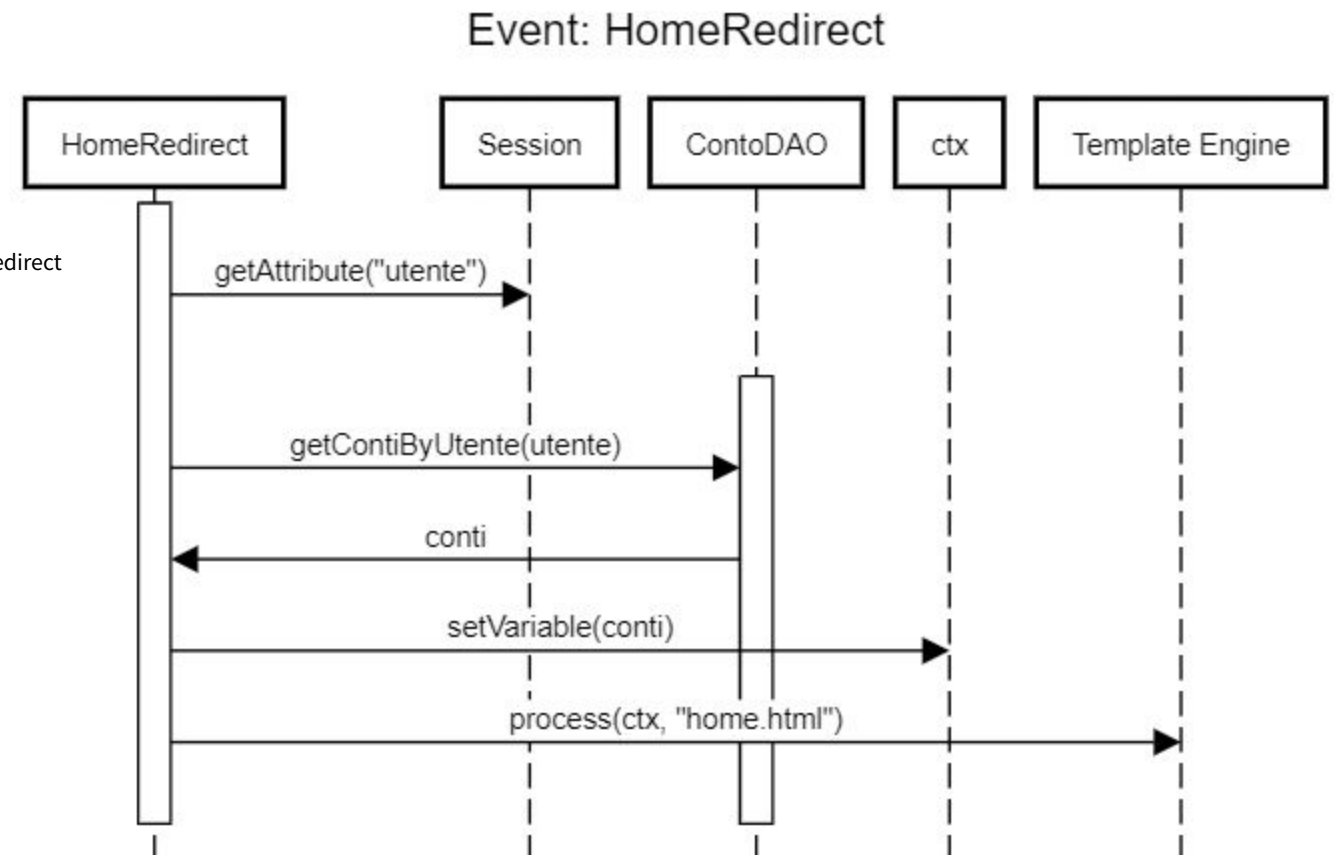
Sequence diagram - HTML



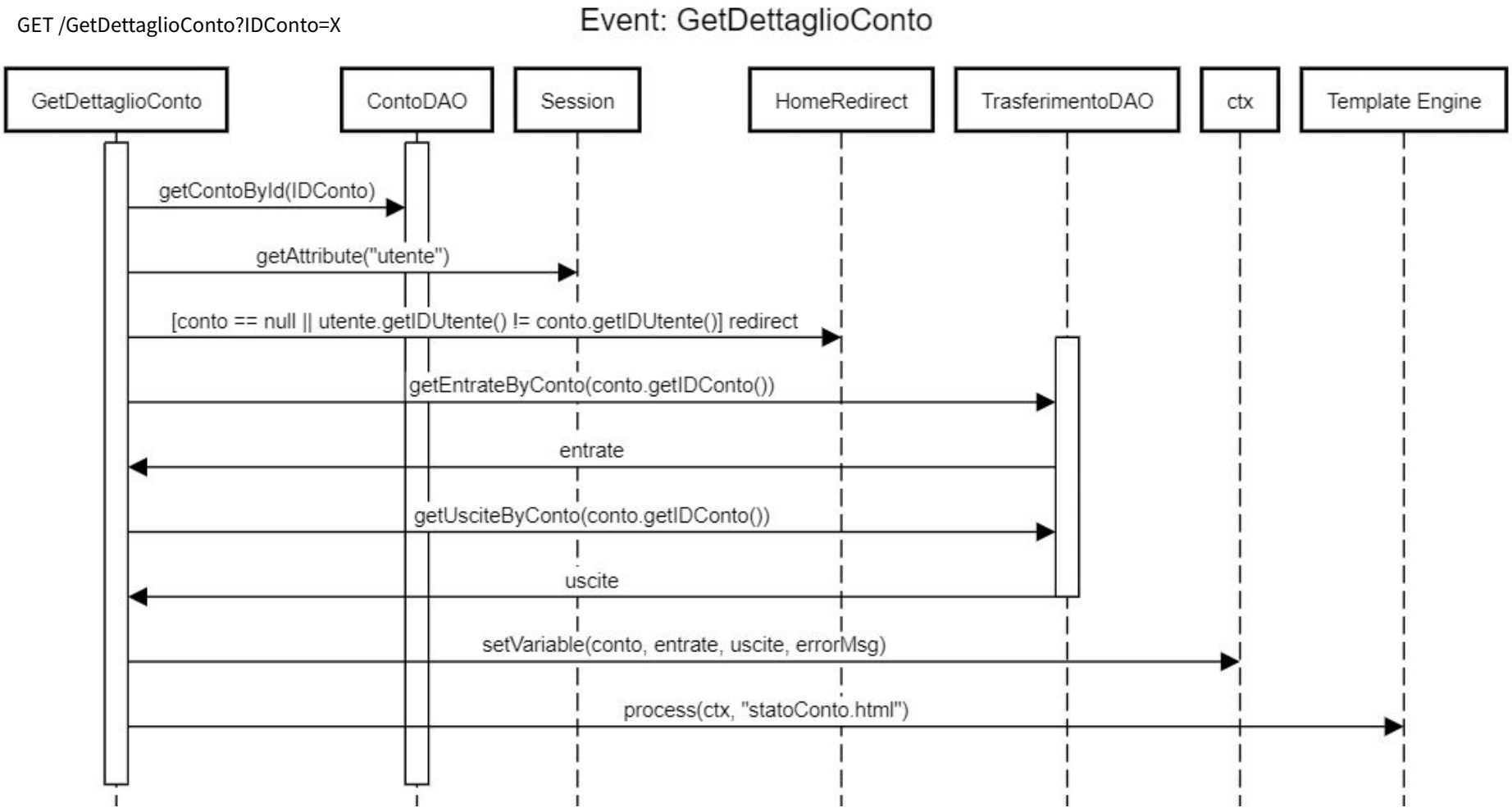
POST (register.html)
nome
cognome
email
password1
password2



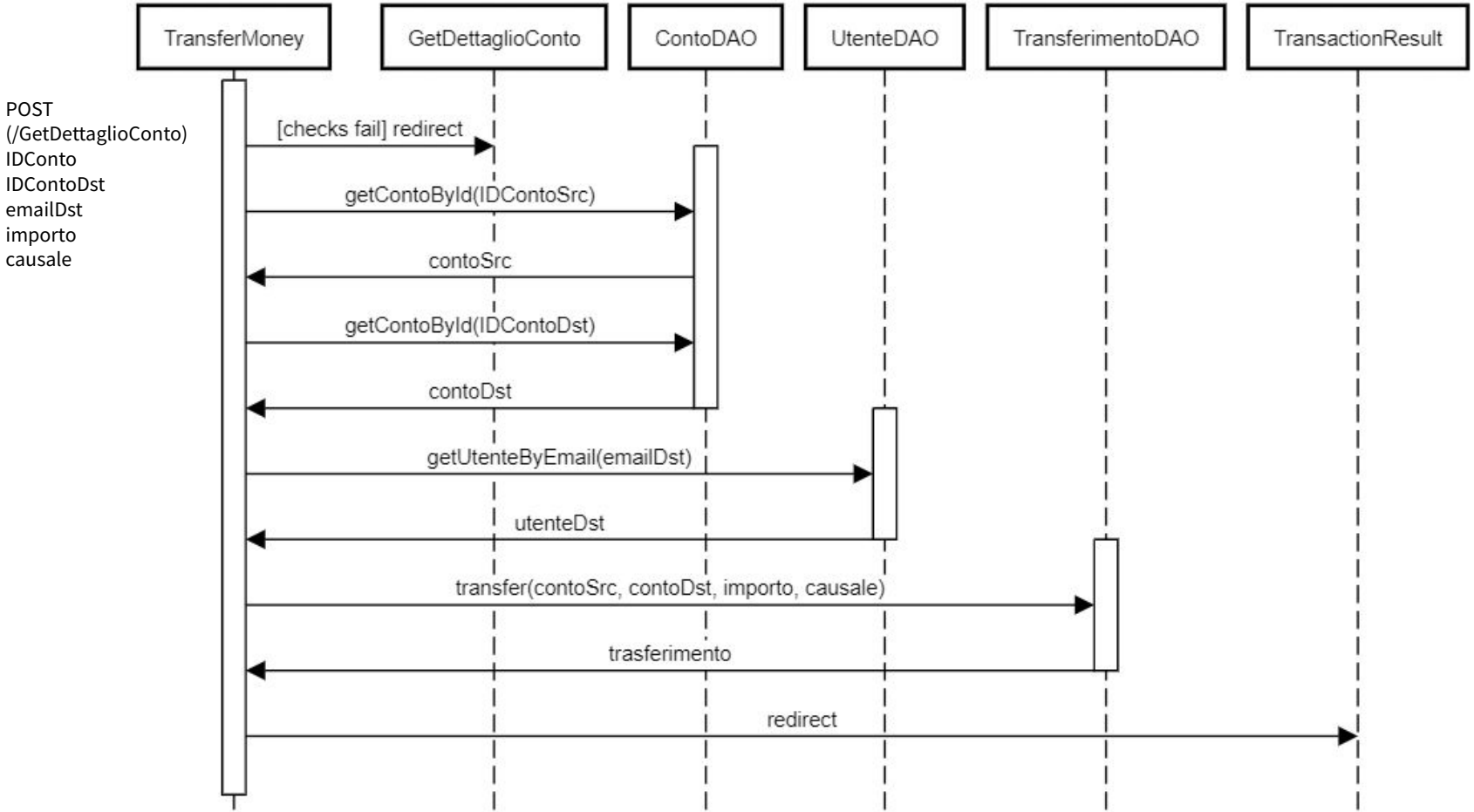
GET /HomeRedirect



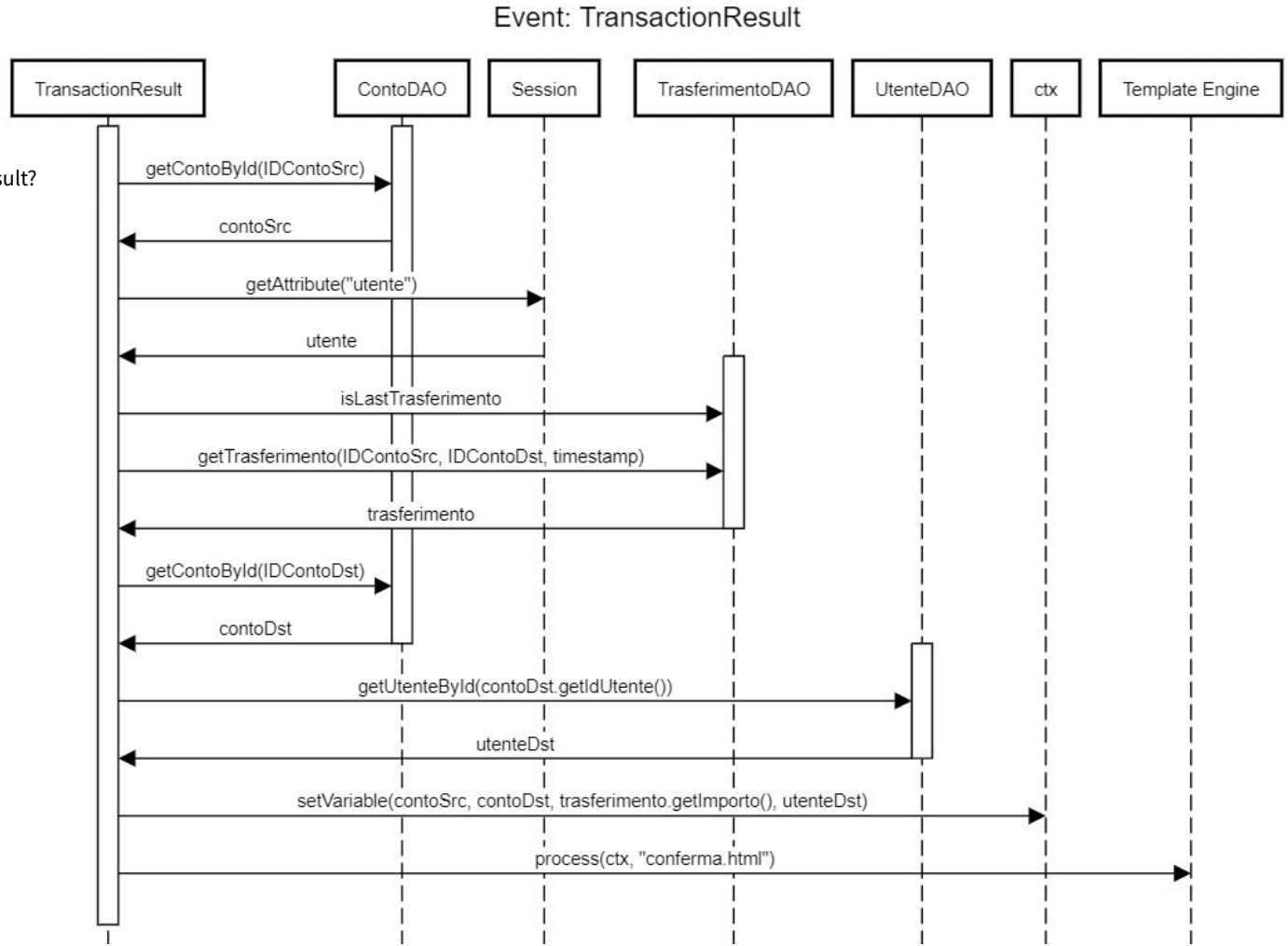
GET /GetDettaglioConto?IDConto=X

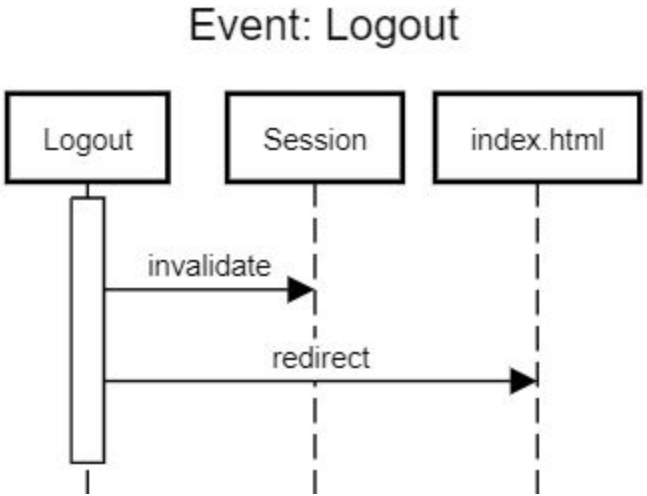


TransferMoney

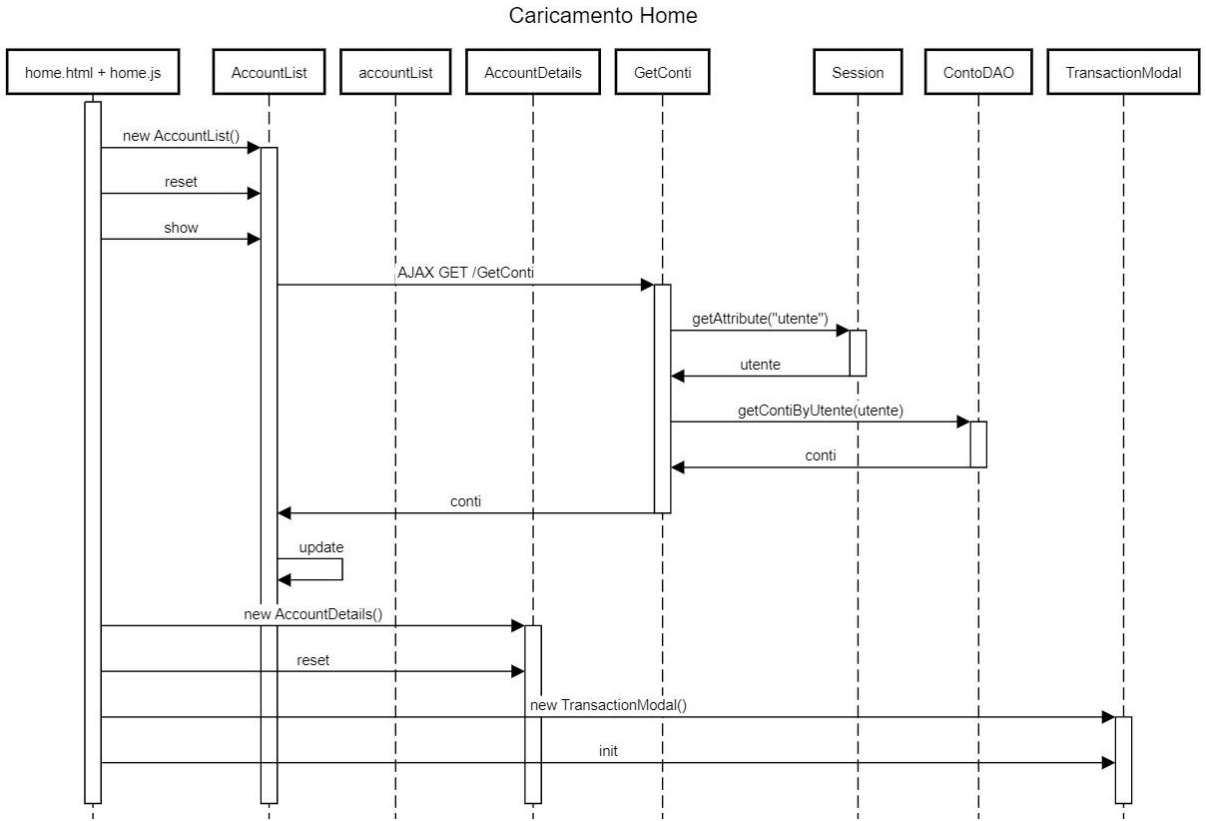


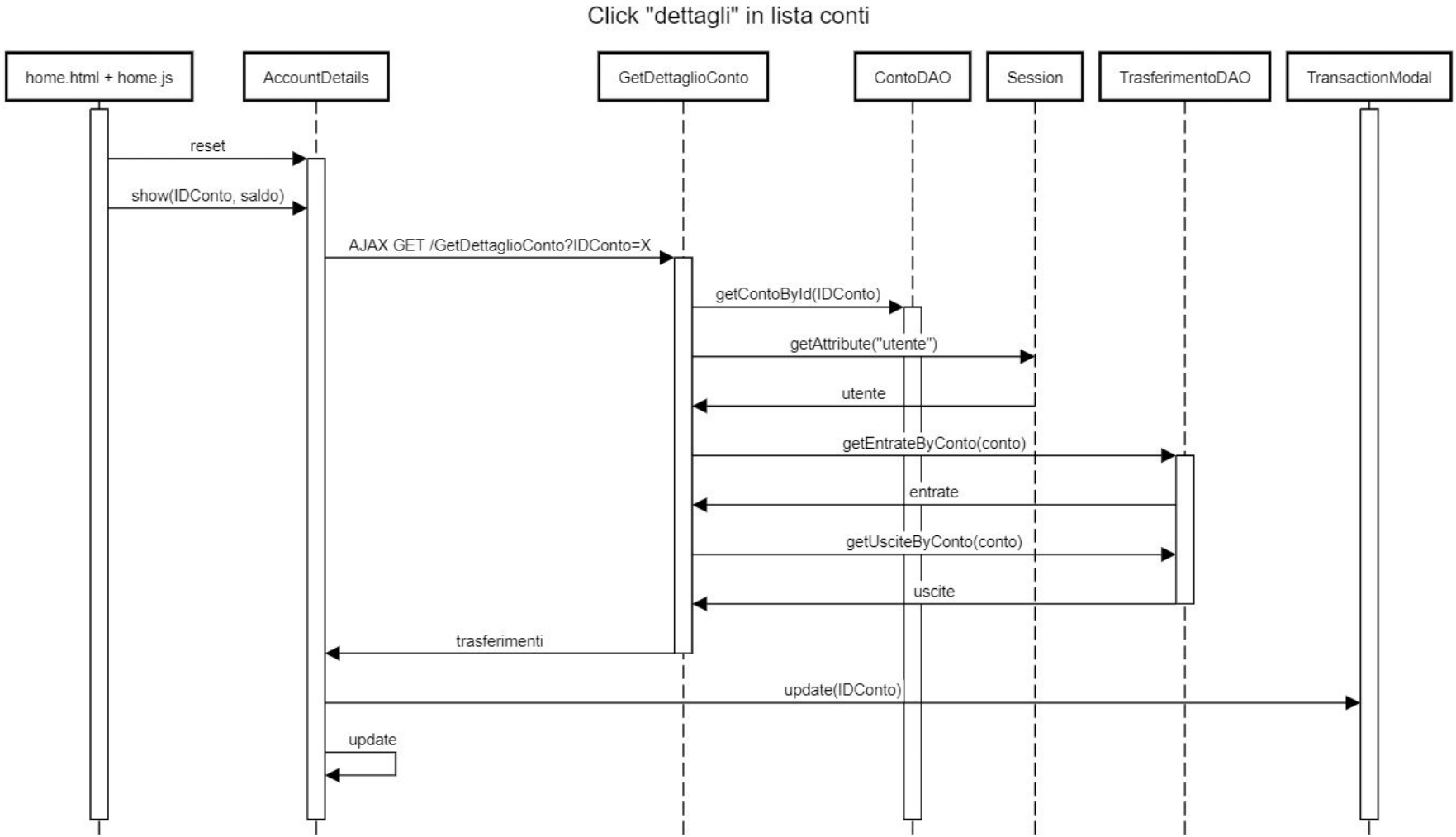
GET /TransactionResult?
IDContoSrc=X&
IDContoDst=Y&
Data=Z

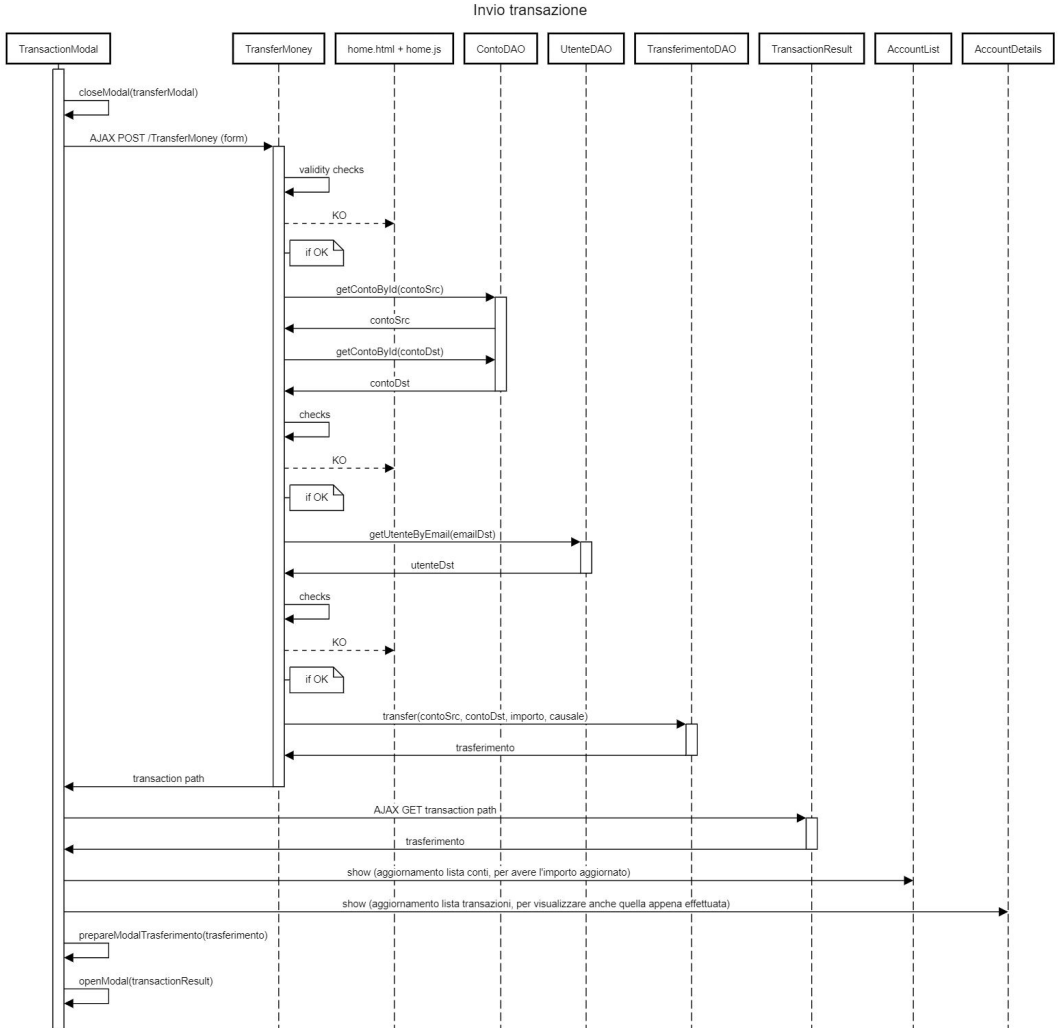


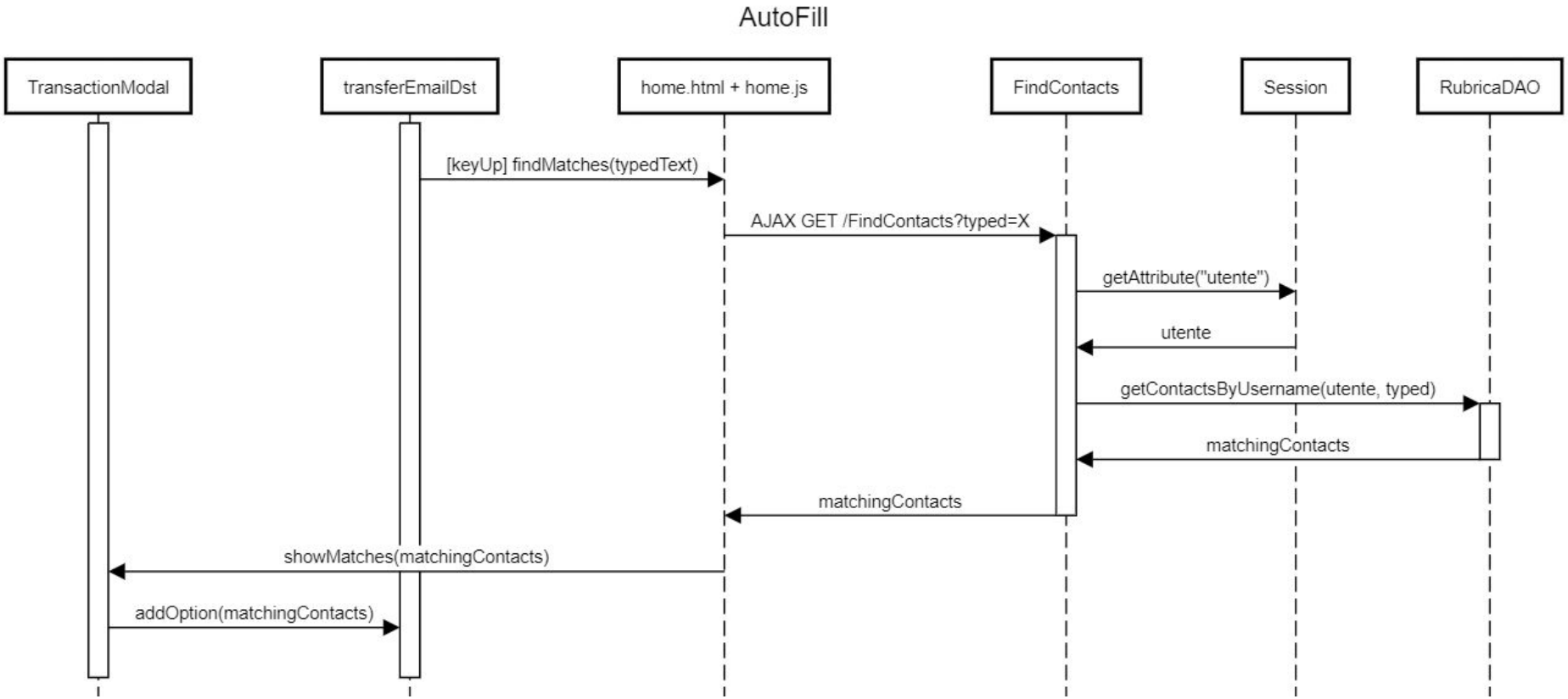

Event: Logout
Logout
Session
index.html
invalidate
redirect

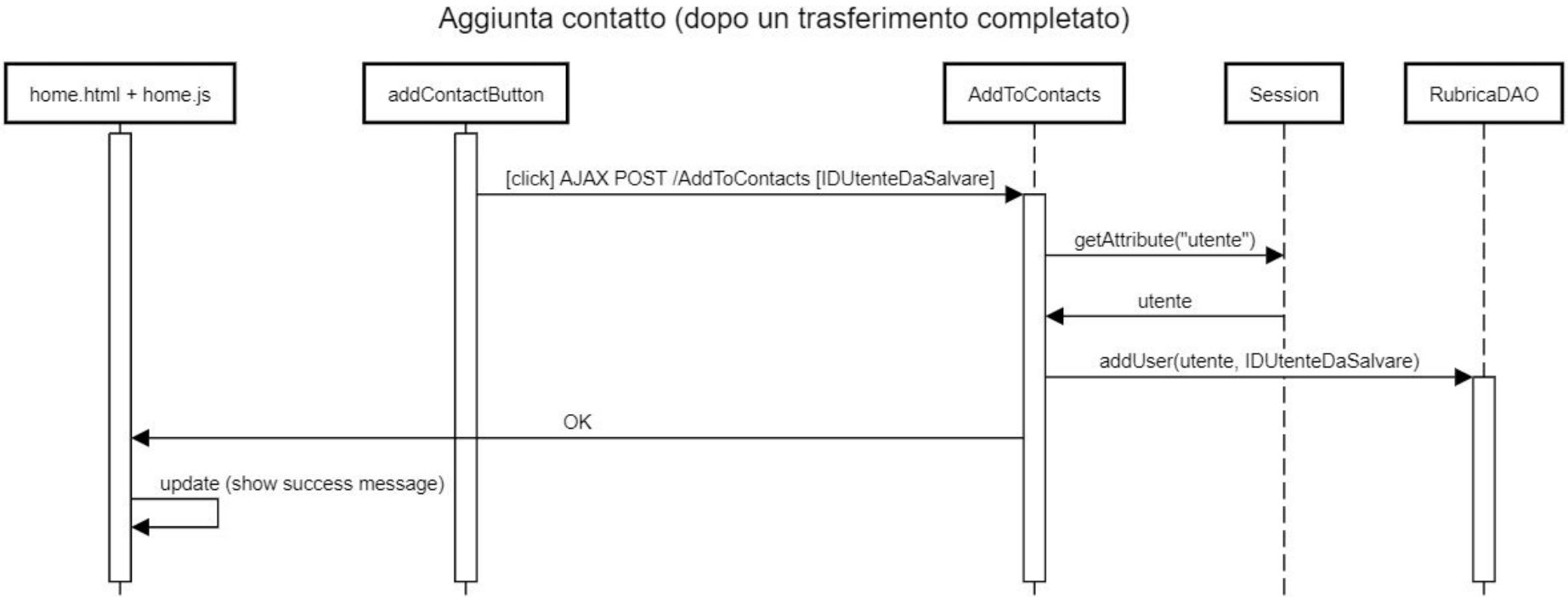
Sequence diagram - RIA



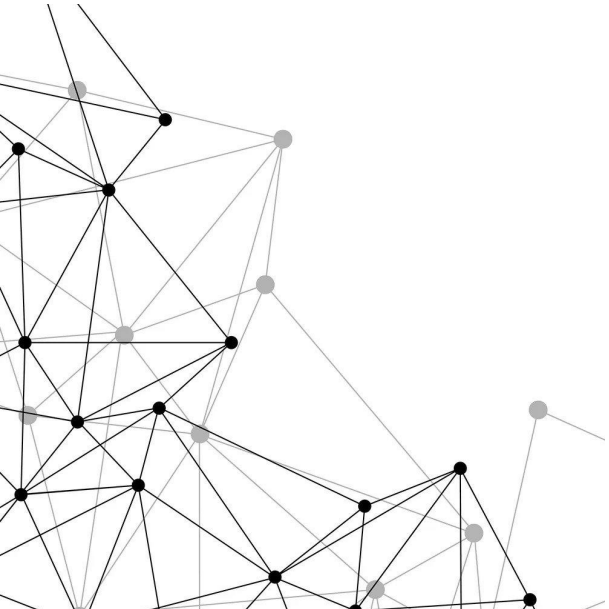

Click "dettagli" in lista conti
home.html + home.js
AccountDetails
GetDettaglioConto
ContoDAO
Session
TrasferimentoDAO
TransactionModal
reset
show(IDConto, saldo)
AJAX GET /GetDettaglioConto?IDConto=X
getContoById(IDConto)
getAttribute("utente")
utente
getEntrateByConto(conto)
entrate
getUsciteByConto(conto)
uscite
trasferimenti
update(IDConto)
update

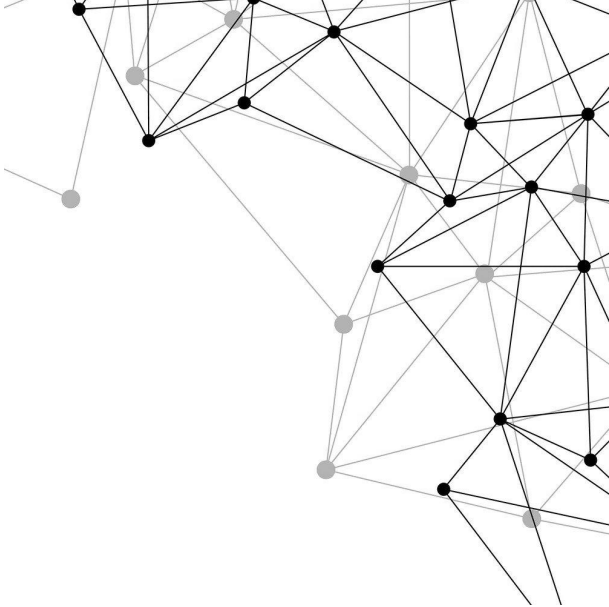

Invio transazione
TransactionModal
TransferMoney
home.html + home.js
ContoDAO
UtenteDAO
TransferimentoDAO
TransactionResult
AccountList
AccountDetails
closeModal(transferModal)
AJAX POST /TransferMoney (form)
validity checks
KO
if OK
getContoById(contoSrc)
contoSrc
getContoById(contoDst)
contoDst
checks
KO
if OK
getUtenteByEmail(emailDst)
utenteDst
checks
KO
if OK
transfer(contoSrc, contoDst, importo, causale)
trasferimento
transaction path
AJAX GET transaction path
trasferimento
show (aggiornamento lista conti, per avere l'importo aggiornato)
show (aggiornamento lista transazioni, per visualizzare anche quella appena effettuata)
prepareModalTrasferimento(trasferimento)
openModal(transactionResult)


AutoFill
TransactionModal
transferEmailDst
home.html + home.js
FindContacts
Session
RubricaDAO
[keyUp] findMatches(typedText)
AJAX GET /FindContacts?typed=X
getAttribute("utente")
utente
getContactsByUsername(utente, typed)
matchingContacts
matchingContacts
showMatches(matchingContacts)
addOption(matchingContacts)


Aggiunta contatto (dopo un trasferimento completato)
home.html + home.js
addContactButton
AddToContacts
Session
RubricaDAO
[click] AJAX POST /AddToContacts [IDUtenteDaSalvare]
getAttribute("utente")
utente
addUser(utente, IDUtenteDaSalvare)
OK
update (show success message)

POLITECNICO
MILANO 1863





FINE