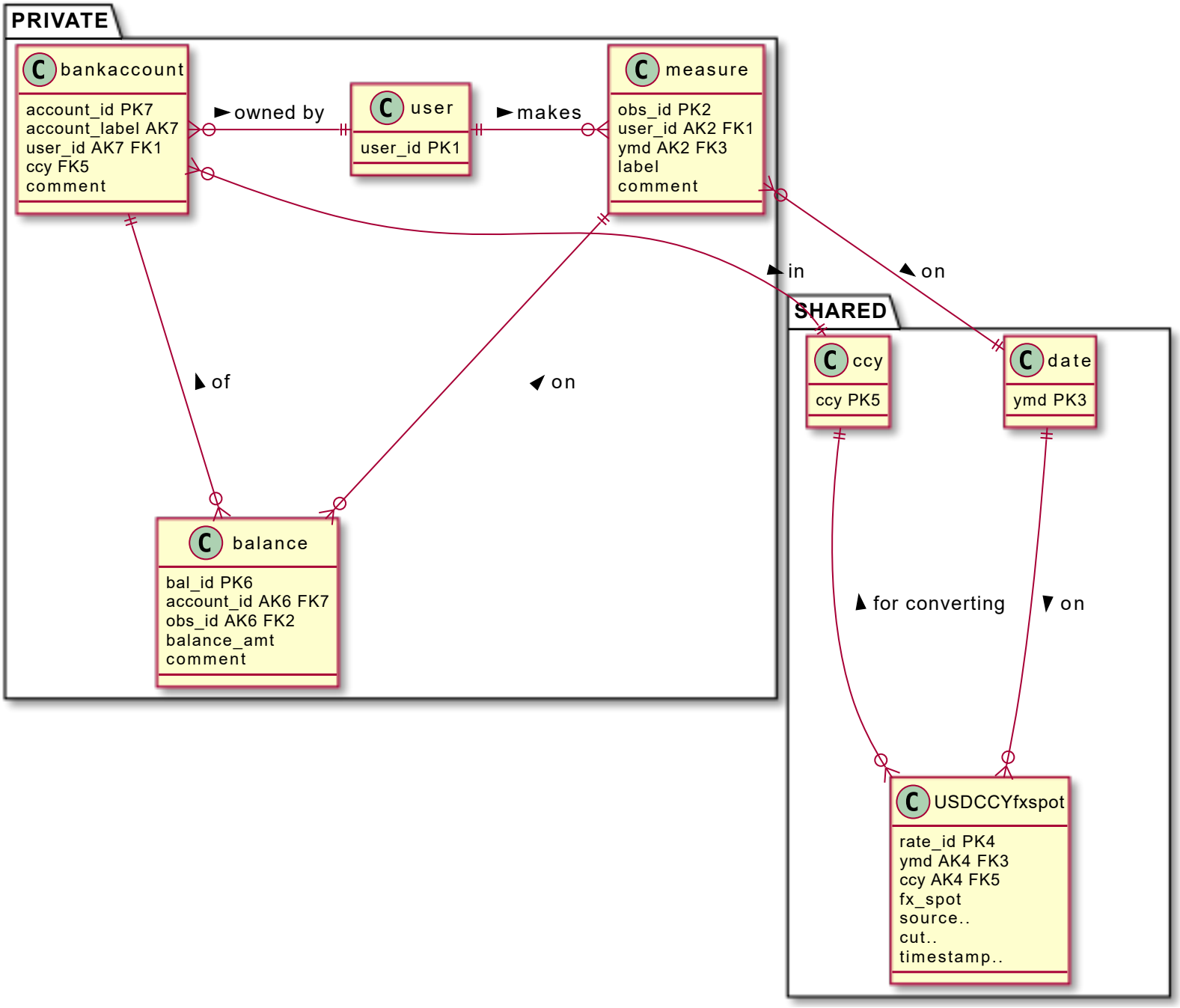
multiccy - blueprints and code

Class diagram of the multiccy app database

- user owns multiple accounts in multiple currencies
- user checks the balance of all the owned accounts on the same date = measure date


PRIVATE
bankaccount
account_id PK7
account_label AK7
user_id AK7 FK1
ccy FK5
comment
owned by
user
user_id PK1
makes
measure
obs_id PK2
user_id AK2 FK1
ymd AK2 FK3
label
comment
in
on
of
on
balance
bal_id PK6
account_id AK6 FK7
obs_id AK6 FK2
balance_amt
comment
SHARED
ccy
ccy PK5
date
ymd PK3
for converting
on
USDCCYfxspot
rate_id PK4
ymd AK4 FK3
ccy AK4 FK5
fx_spot
source..
cut..
timestamp..

Value-at-Risk function

```
// groovy

    Double get_1d95var(String ccy, List rateHistory, Double histVaRPctile, Double latestAmountAtRiskCcy) {

        if(ccy=="EUR") return 0.0
        assert histVaRPctile==0.95 // || histVaRPctile==0.99
        // GIVEN rateHistory [0.902, 0.904, 0.903, 0.906, 0.904]

        Map latestRateMap = get_1d95var_api_latest(ccy)    // can be null. unaddressed. FT20201018c
        Double latestRate = latestRateMap?.rates[ccy].toDouble()



        int isz = rateHistory.size()    // isz = 5
        int dsz = rateHistory.size()-1    // dsz = 5-1

        def subs = 1..dsz
        List runningHistRateDelta = subs.collect { rateHistory[it] - rateHistory[it-1]}
        // THEN runningHistRateDelta [0.002, -0.001, 0.003, -0.002]
        // GIVEN latestRate 0.907    // not necessarily = rateHistory[isz-1]
        // GIVEN latestAmountAtRiskCcy 1000.0

        Double latestAmountEur = latestAmountAtRiskCcy / latestRate
        // THEN latestAmountEur = 1102.5358

        List EstimatedTomorrowRateSet = runningHistRateDelta.collect { latestRate + it }
        // THEN EstimatedTomorrowRateSet [0.909, 0.906, 0.91, 0.905]

        // revaluation:
        List EstimatedTomorrowAmountEur = EstimatedTomorrowRateSet.collect { latestAmountAtRiskCcy / it }
```

```
        // THEN EstimatedTomorrowAmountEur [1100.1100, 1103.7527, 1098.9010, 1104.9723]

        List EstimatedTomorrowAmountEurSorted = EstimatedTomorrowAmountEur.sort().reverse() // descending
        // THEN EstimatedTomorrowAmountEurSorted [1104.9723, 1103.7527, 1100.1100, 1098.9010]
        int pctile = Math.ceil((dsz-1) * histVaRPctile).toInteger()
        Double var95_EstimatedAmount = EstimatedTomorrowAmountEurSorted[pctile]
        // THEN VaR = var95_EstimatedAmount - latestAmountEur = 1098.9010 - 1102.5358 = -3.6348 EUR
        return var95_EstimatedAmount - latestAmountEur // = var95_EstimatedLoss
    }
```

delta report by ccy (in Cypher, for graph DBs)

```
WITH timestamp() as TS match (uu:USER {nm:\$tkusn}) with uu, TS
        match (uu)-[ud:HASMANY]->(ddz:DATE)
        with uu,
            max(ddz.dt) as date2,
            '1' as rowcount
        match (uu)-[ud:HASMANY]->(dd:DATE) where dd.dt = date2
        match (uu)<-[ua:HASONE]-(aa:ACCOUNT)<-[ab:HASONE]-(bb:BALANCE)-[bd:HASONE]->(dd)
        match (aa)-[ac:HASONE]->(cc:CURRENCY)<-[cx:HASONE]-(eurccy:HISTFX)-[dx:HASONE]->(dd)

        with uu, date2, cc.nm as ccy2,
            sum(bb.val/eurccy.rate) as eur2_byccy,
            collect(eurccy.rate)[0] as rate2,
            sum(bb.val) as amt2_byccy,
            '1*ccy' as rowcount
        match (uu)-[ud:HASMANY]->(dd:DATE) where dd.dt < date2
        match (uu)<-[ua:HASONE]-(aa:ACCOUNT)<-[ab:HASONE]-(bb:BALANCE)-[bd:HASONE]->(dd)
        match (aa)-[ac:HASONE]->(cc:CURRENCY)<-[cx:HASONE]-(eurccy:HISTFX)-[dx:HASONE]->(dd) where cc.nm=ccy2

        with date2, ccy2, eur2_byccy, rate2, amt2_byccy,
             dd.dt as date1, cc.nm as ccy1,
```

```
            sum(bb.val/eurccy.rate) as eur1_byccy,
            collect(eurccy.rate)[0] as rate1,
            sum(bb.val) as amt1_byccy,
            '1*ccy*(dt-1)' as rowcount

        with date2, ccy2, eur2_byccy, rate2, amt2_byccy,
            date1, ccy1, eur1_byccy, rate1, amt1_byccy,
            eur2_byccy - eur1_byccy as delta_eur22_eur11_byccy,
            (amt2_byccy / rate1) - eur1_byccy as delta_eur21_eur11_byccy,
            (amt1_byccy / rate2) - eur1_byccy as delta_eur12_eur11_byccy,
            '1*ccy*(dt-1)' as rowcount
        with date1, date2, sum(eur2_byccy) as eur2,
        sum(delta_eur22_eur11_byccy) as delta_eur22_eur11,
        sum(delta_eur21_eur11_byccy) as delta_eur21_eur11,
        sum(delta_eur12_eur11_byccy) as delta_eur12_eur11,
        sum(eur1_byccy) as eur1,
        '1*(dt-1)' as rowcount
        return {
        id:      date1,
        vm31dt:  date1,
        vm31dtz: date2,
        vm31ab:  eur1,
        vm31abz: eur2,
        vm31abd: delta_eur22_eur11,
        vm31abp: 100.0 * delta_eur22_eur11 / eur1,
        vm31abdfzfx: delta_eur21_eur11,
        vm31abpfzfx: 100.0 * delta_eur21_eur11 / eur1
        } order by date1
```

delta report by ccy (equivalent in SQL, for relational DBs)

```
CREATE VIEW ccydt_matrix as
```

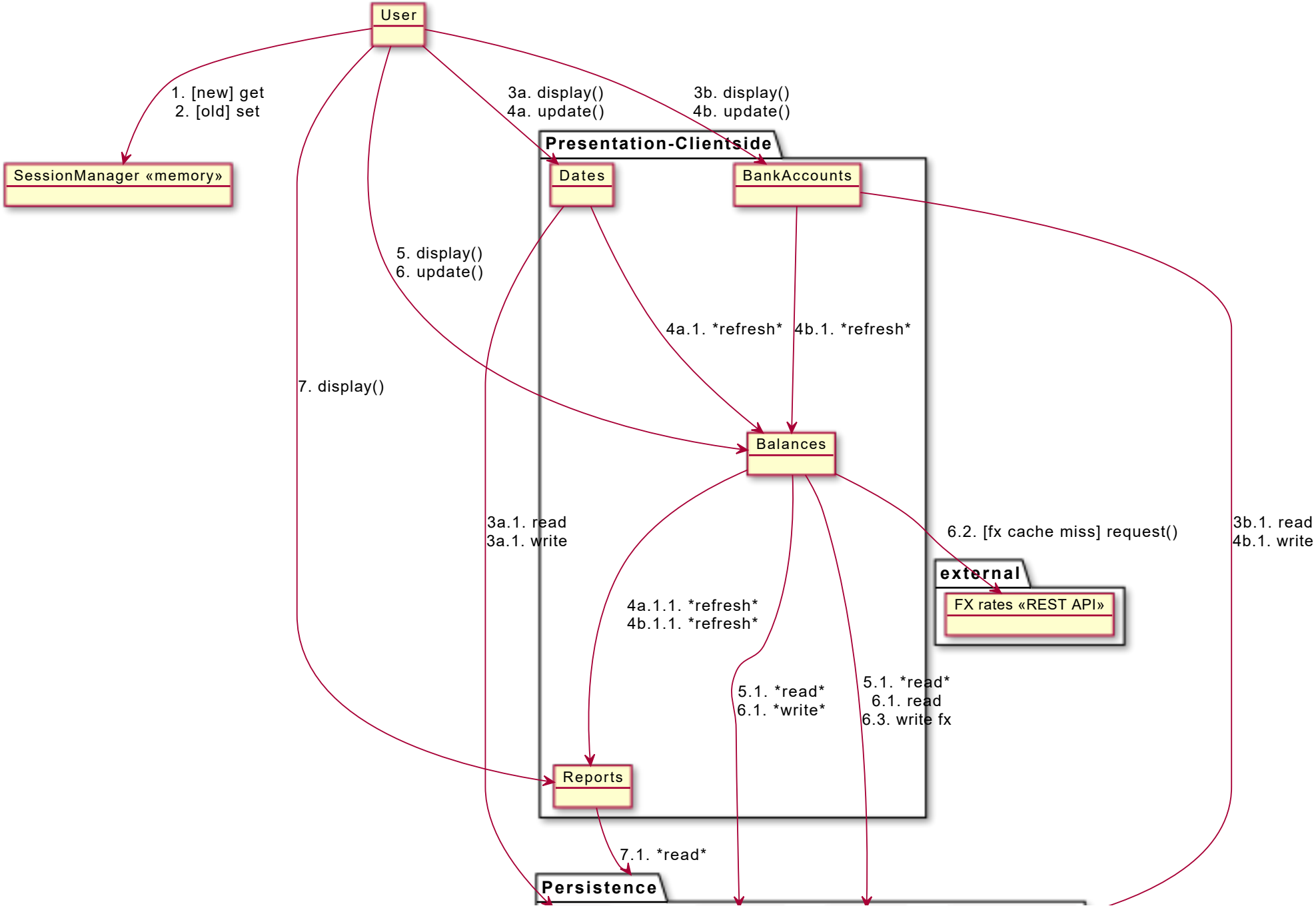
```
select ymd,ccy from dates, (select distinct ccy from accounts)
-- expected rowcount: currencies * dates

CREATE VIEW balances_eur as
select balances.ymd, accounts.account, accounts.ccy, xrates.eur as rate,
balances.amount, balances.amount / xrates.eur as amount_eur
from accounts, balances, xrates
where accounts.account = balances.account and xrates.ymd=balances.ymd
and accounts.ccy = xrates.ccy

CREATE VIEW balances_eur_bydtccy as select ymd, ccy, rate,
sum(amount) as amount, sum(amount_eur) as amount_eur
from balances_eur
group by ymd, ccy, rate

CREATE VIEW balances_eur_bydtccy_matrix as
select ccydt_matrix.ccy, ccydt_matrix.ymd, balances_eur_bydtccy.rate,
coalesce(balances_eur_bydtccy.amount,0) as amount,
coalesce(balances_eur_bydtccy.amount_eur,0) as amount_eur
from ccydt_matrix left outer join balances_eur_bydtccy
on ccydt_matrix.ymd = balances_eur_bydtccy.ymd
and ccydt_matrix.ccy = balances_eur_bydtccy.ccy
-- expected rowcount: currencies * dates

CREATE VIEW balances_eur_bydtccy_matrix12 as
select md1.ccy, md1.ymd, md1.rate, md1.amount, md1.amount_eur,
md2.ymd as ymd2, md2.rate as rate2, md2.amount as amount2,
md2.amount_eur  as amount_eur2,
coalesce(md2.amount / md1.rate,0) as amount_eur2nofx
from
balances_eur_bydtccy_matrix md2, balances_eur_bydtccy_matrix md1
where md2.ymd = (select max(ymd) from dates)
and md1.ccy = md2.ccy and md1.ymd < md2.ymd
-- expected rowcount: currencies * dates-1
```

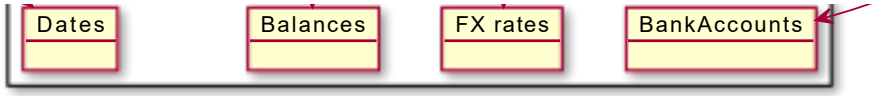
```
-- (dates are stored in field ymd as a yyyy-mm-dd 10-char string)

CREATE VIEW delta_report_nofx as select
ymd, sum(amount_eur) amount_eur,
ymd2, sum(amount_eur2) amount_eur2,
sum((amount_eur2 - amount_eur)) as diff_eur,
100.0*sum((amount_eur2 - amount_eur)/amount_eur) as diffpct_eur,
sum((amount_eur2nofx - amount_eur)) as diff_eur_nofx,
100.0*sum((amount_eur2nofx - amount_eur)/amount_eur) as diffpct_eur_nofx
from balances_eur_bydtccy_matrix12
group by ymd, ymd2
-- expected rowcount: dates-1

select * from delta_report_nofx
```

GUI events - eager-loading vs. lazy-loading models

eager-loading model

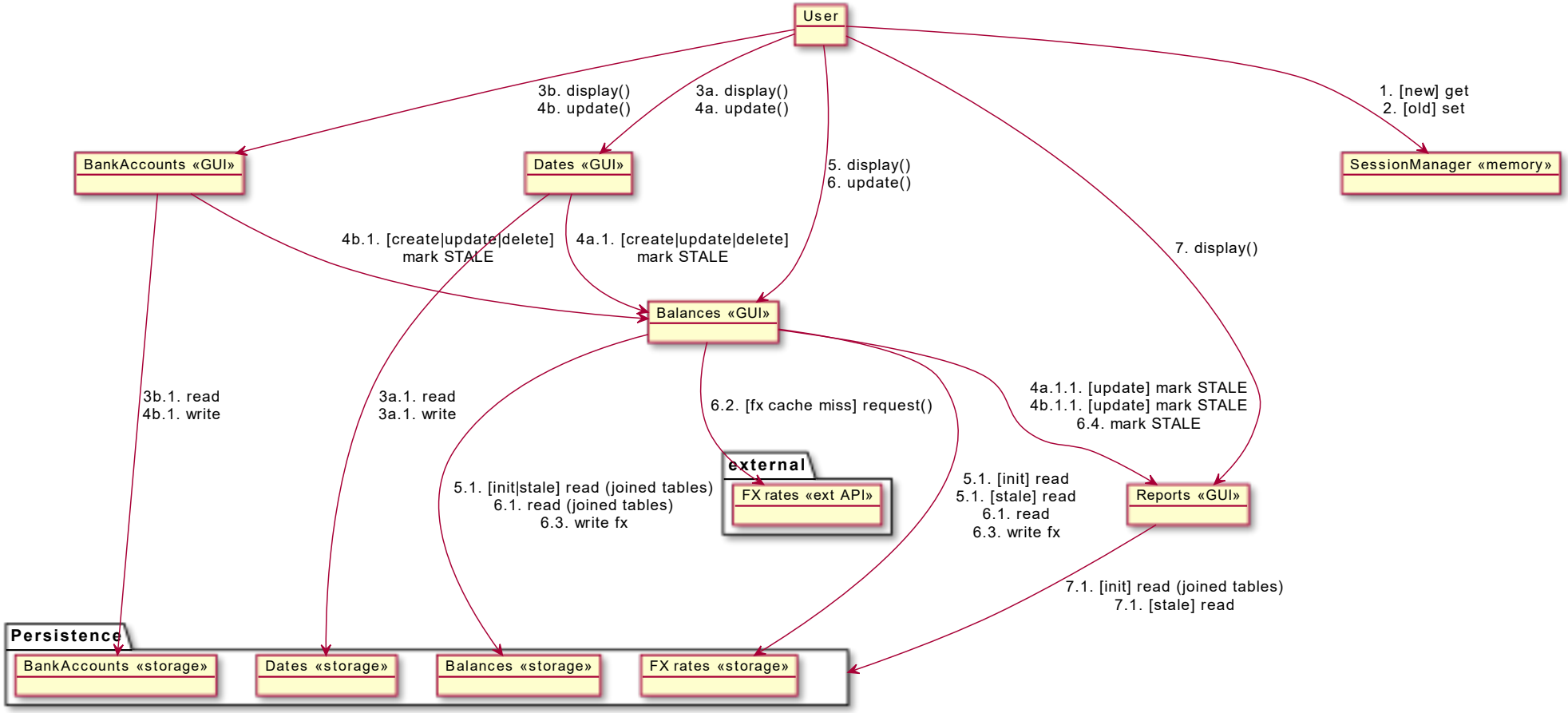
- maximizes the requests sent to the server.
- decrease the waiting time of the user.


User
1. [new] get
2. [old] set
SessionManager «memory»
3a. display()
4a. update()
3b. display()
4b. update()
Presentation-Clientside
Dates
BankAccounts
5. display()
6. update()
4a.1. *refresh*
4b.1. *refresh*
7. display()
Balances
3a.1. read
3a.1. write
6.2. [fx cache miss] request()
3b.1. read
4b.1. write
external
FX rates «REST API»
4a.1.1. *refresh*
4b.1.1. *refresh*
5.1. *read*
6.1. *write*
5.1. *read*
6.1. read
6.3. write fx
Reports
7.1. *read*
Persistence



lazy-loading model

- minimizes the requests sent to the server.
- increase the waiting time of the user.

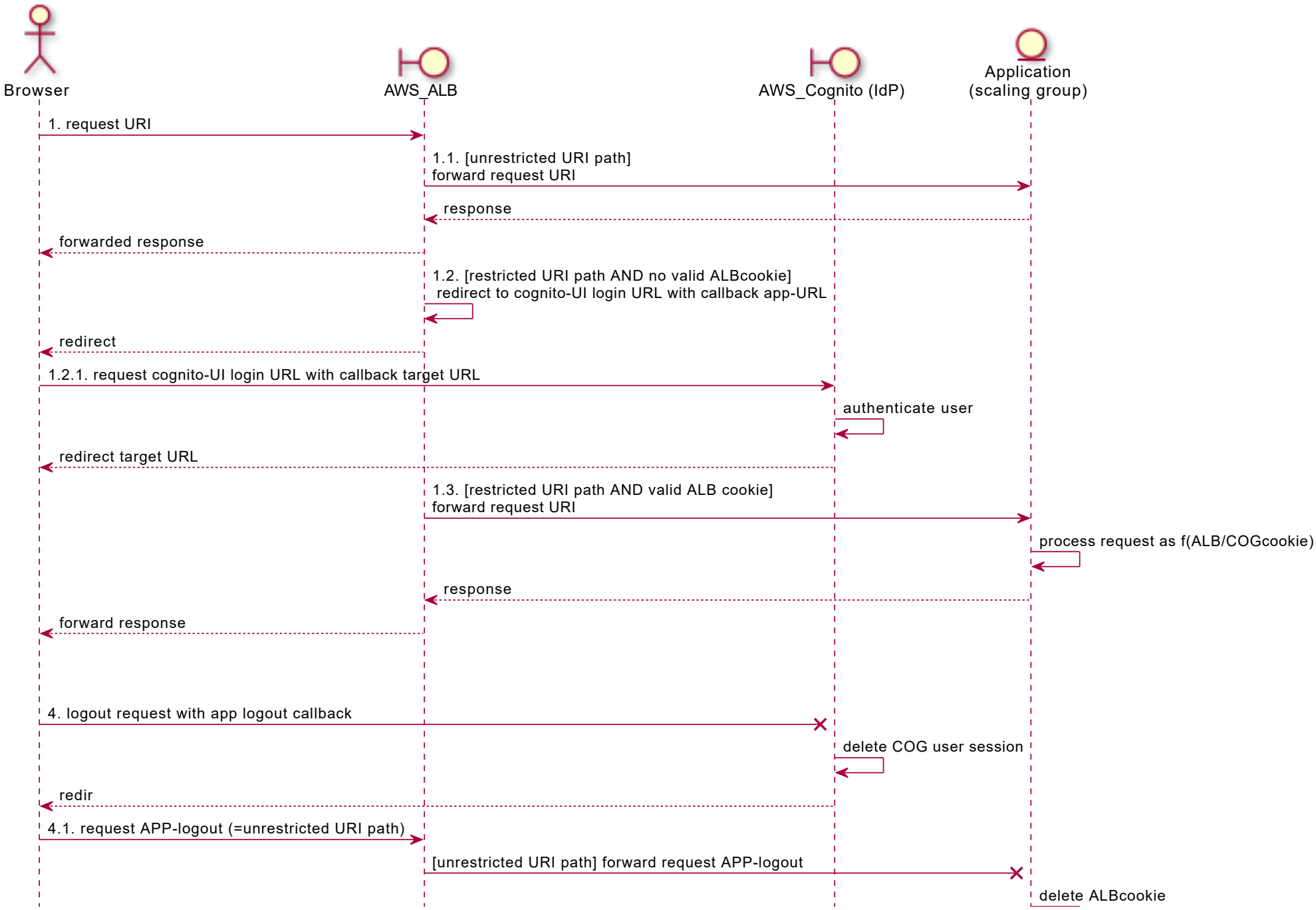


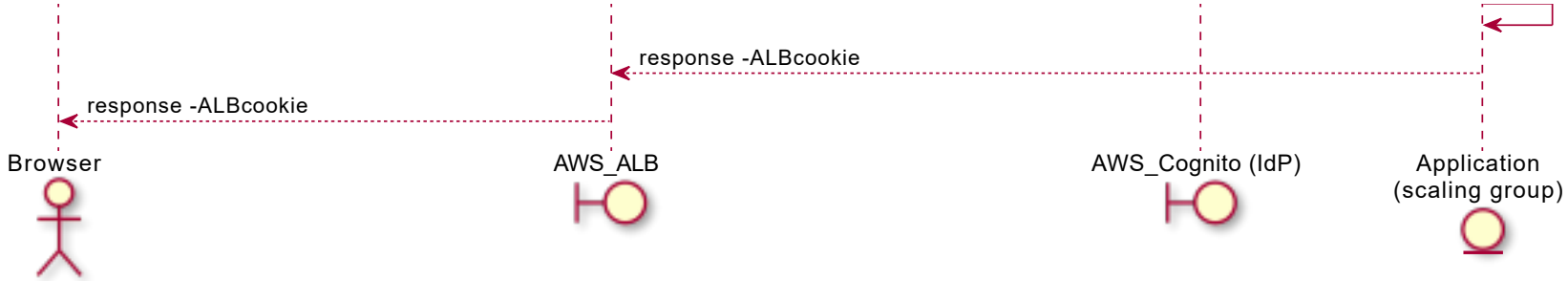
AWS Cognito hosted-UI integration

Below is the sequence diagram of the http requests between your browser and
the application multiccy .
The IAM is managed by the combo AWS ALB + AWS Cognito IdP + AWS Cognito hosted UI.
With this integration, your application
does NOT have to:

- keep a users database;
- implement sign in, sign up, password reset services and emailing;
- use a session-management database like Redis/Dynamodb;
  should:
- be using the http header 'X-Amzn-Oidc-Identity', which is the authenticated cognito user id, as a session userid
- delete the AWSELBAuthSessionCookie in the logout callback URL call

simplified sequence diagram:


Browser
AWS_ALB
AWS_Cognito (IdP)
Application
(scaling group)
1. request URI
1.1. [unrestricted URI path]
forward request URI
response
forwarded response
1.2. [restricted URI path AND no valid ALBcookie]
redirect to cognito-UI login URL with callback app-URL
redirect
1.2.1. request cognito-UI login URL with callback target URL
authenticate user
redirect target URL
1.3. [restricted URI path AND valid ALB cookie]
forward request URI
process request as f(ALB/COGcookie)
response
forward response
4. logout request with app logout callback
delete COG user session
redir
4.1. request APP-logout (=unrestricted URI path)
[unrestricted URI path] forward request APP-logout
delete ALBcookie



see detailed AUTH sequence diagram

ref. https://docs.aws.amazon.com/elasticloadbalancing/latest/application/listener-authenticate-users.html - the numbers of the workflow are like (#1) above

changes made to the application to enable the AWS Cognito hosted-UI integration

gains/pains:

- gains:
  - no need to custom develop the registration process (pages, emails, password reset, password policy enforcement);
  - little changes to be implemented in the application;
  - no need to use a session management database like redis/memcache/dynamodb to support scaling-out.
- pains:
  - lost ownership of the user directory.
  - Cost.

key points:

- the landing page "/" is the public page /public/index rendered by the public controller /public
- the landing page shows a Login link to the private homepage /member/index rendered by an authenticated controller
- the login is handled by the AWS ALB + AWS Cognito
- the private home can set the application session data upon needs

- the logout link is a request to a /public/start_logout method that knows the COGNITO_LOGOUT_URL to redirect to
- the COGNITO_LOGOUT_URL includes a public method logout4cb that deletes the AWS ALB session cookies and the application session cookies.

```
// MemberController
def index() {
    String cognito_user_id = request.getHeader('X-Amzn-Oidc-Identity')
    String cognito_user_name = utilService.extract_field_from_JWT(request.getHeader('X-Amzn-Oidc-Data'), 'username')

  /*
  // no:
  session.uid = cognito_user_id
    session.uname = cognito_user_name
    */
  if(! cognito_user_name) {
            forward(controller: "public", action:"start_logout")
        } else {
            render(view: "privatehome", model: [uname:cognito_user_name, session_uid:cognito_user_id, ...])
        }
```

```
<html>
  <body>
    <script>
        //on click on Logout link:
    window.location = "/public/start_logout"
        </script>
  </body>
</html>
```

```
// PublicController
// just a convenience URL
```

```
def start_logout() {
    String clurl = System.getenv("COGNITO_LOGOUT_URL")
  redirect(url:clurl)
}

def logout4cb() {
        // callback of the cognito logout.
    // mission: delete the AWS ALB session cookies and the application session cookies.
        request.getCookies().findAll { it.getName().startsWith('AWSELBAuthSessionCookie') }.each { ck ->
            Cookie delendumCookie = new Cookie( ck.getName(), 'deleted' )
            delendumCookie.path = '/'
            delendumCookie.maxAge = 0
            response.addCookie delendumCookie
        }

    /*
    // avoid:
        session.code=null
        session.invalidate()
    */
    redirect(url: "https://multiccy.a-moscatelli.info/")
}


//  COGNITO_LOGOUT_URL = the cognito domain logout URL + the callback
// example:
// https://xxxxxx.amazoncognito.com/logout?client_id=yyyyyy&logout_uri= https://multiccy.a-moscatelli.info/public/lo
```

---

back to app-multiccy