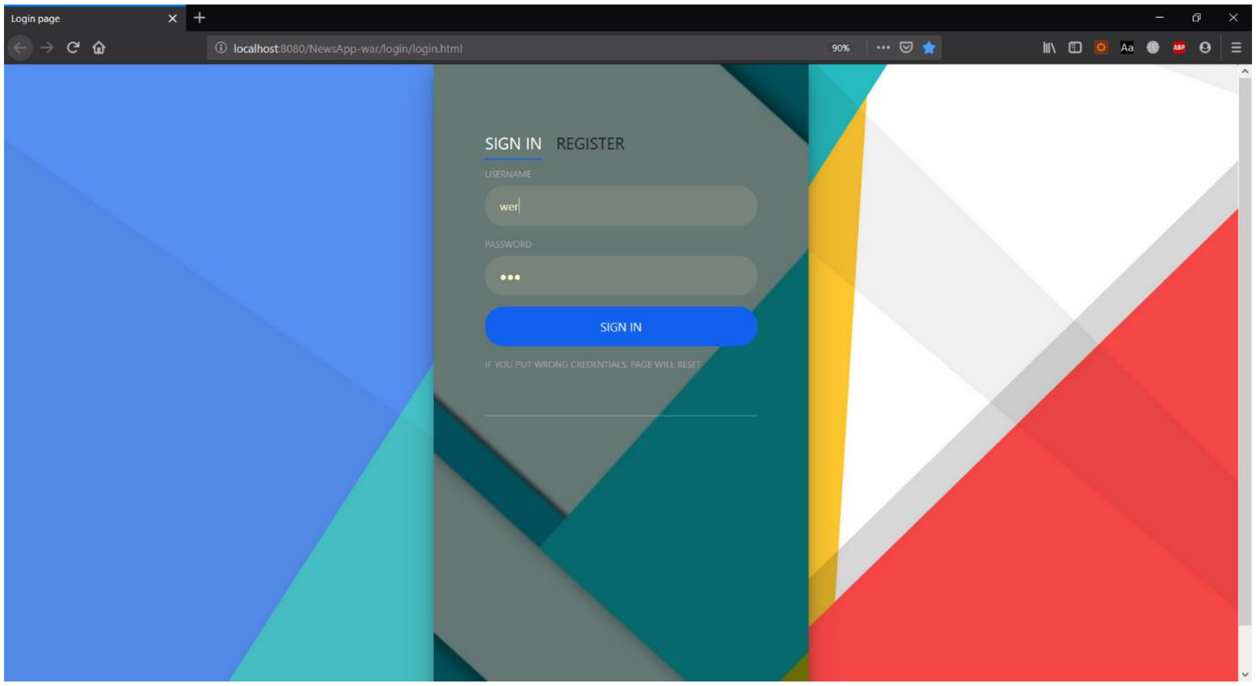
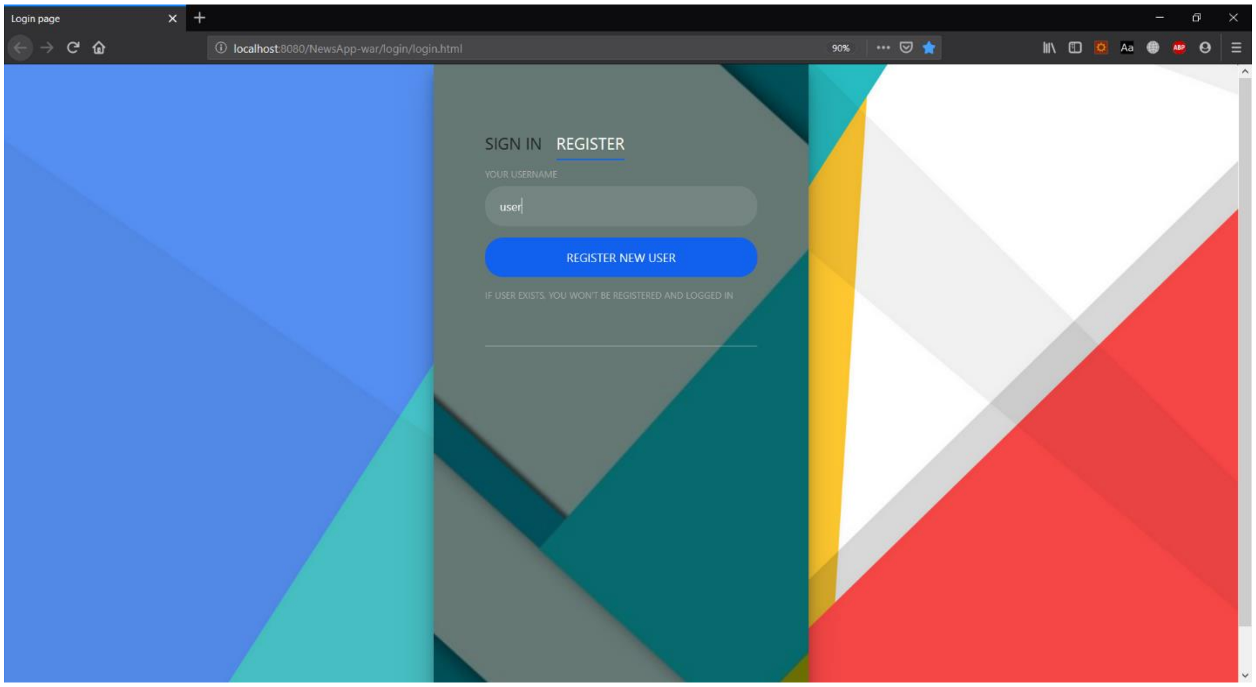
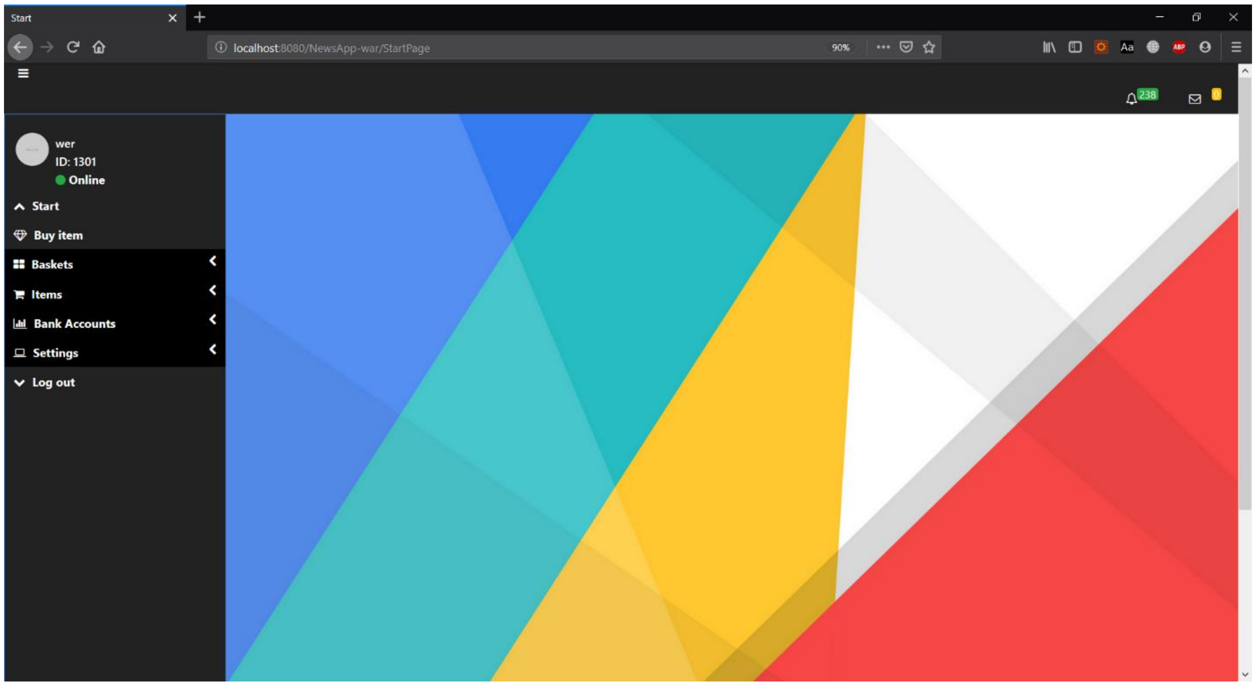
KU LEUVEN – DISTRIBUTED APPLICATION PROJECT – GROCERY-STORE – BARTOSZ SOBKOWIAK – R0768717

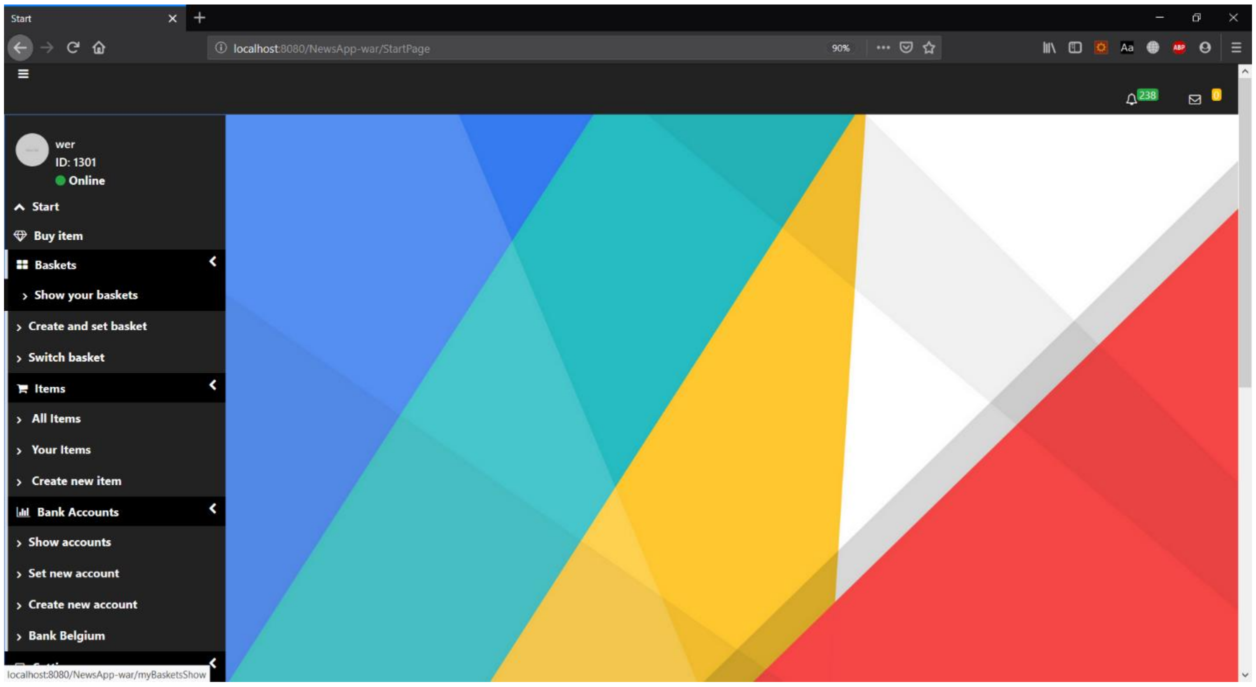

Login page
localhost:8080/NewsApp-war/login/login.html
SIGN IN
REGISTER
USERNAME
wer
PASSWORD
SIGN IN
IF YOU PUT WRONG CREDENTIALS, PAGE WILL RESET

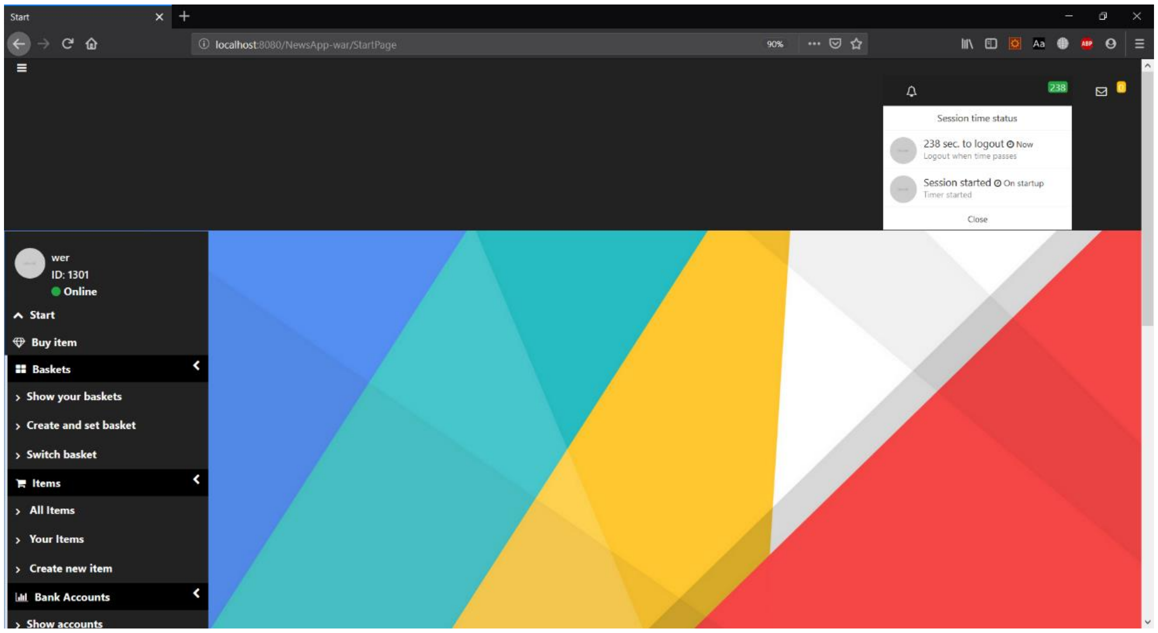
User can sign in with login and password or register as a new user. In the 2nd case password is equal to user's login. If login or password are wrong – page will reload/reset itself. If one wants to register user with name which already exists – page also reload/reset itself.


Login page
localhost:8080/NewsApp-war/login/login.html
SIGN IN
REGISTER
YOUR USERNAME
user
REGISTER NEW USER
IF USER EXISTS, YOU WON'T BE REGISTERED AND LOGGED IN

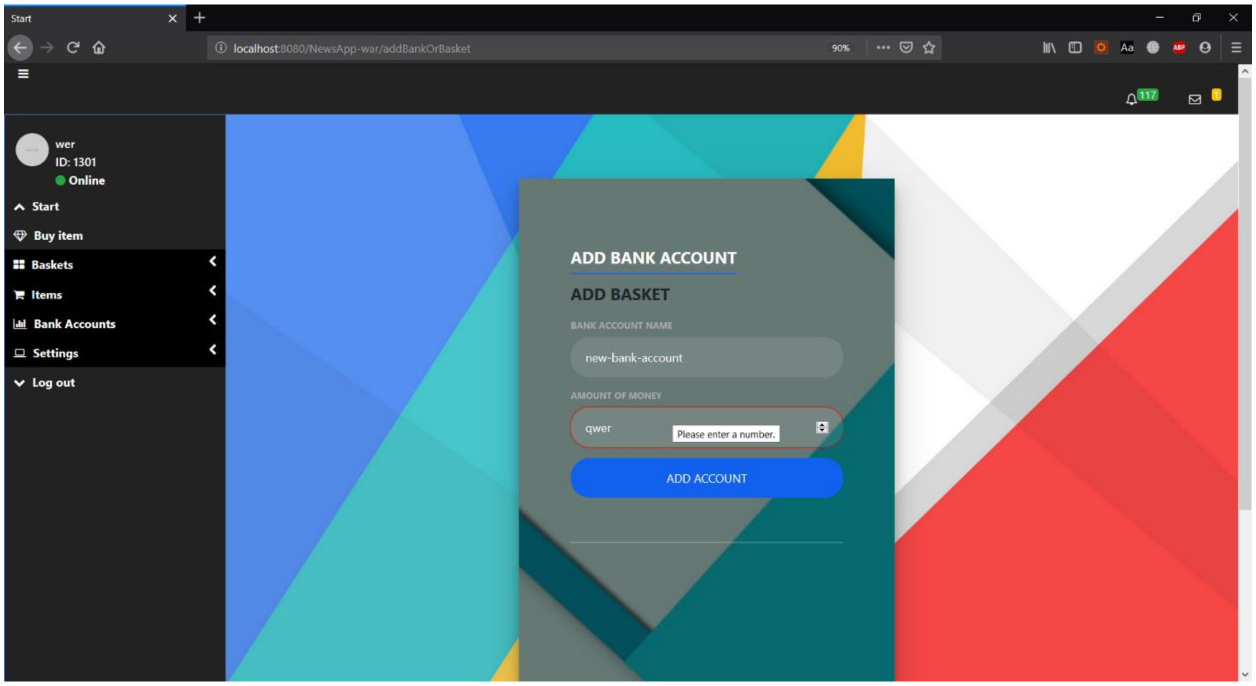


Main screen of an app allows user to choose actions like creating elements, setting them, assigning, modifying, checking status, showing etc. Left menu animation is supported by JavaScript. The JSP page is based on HTML file with Bootstrap templates engine. JSP allows to inject data from EJB beans as current user name, ID etc.

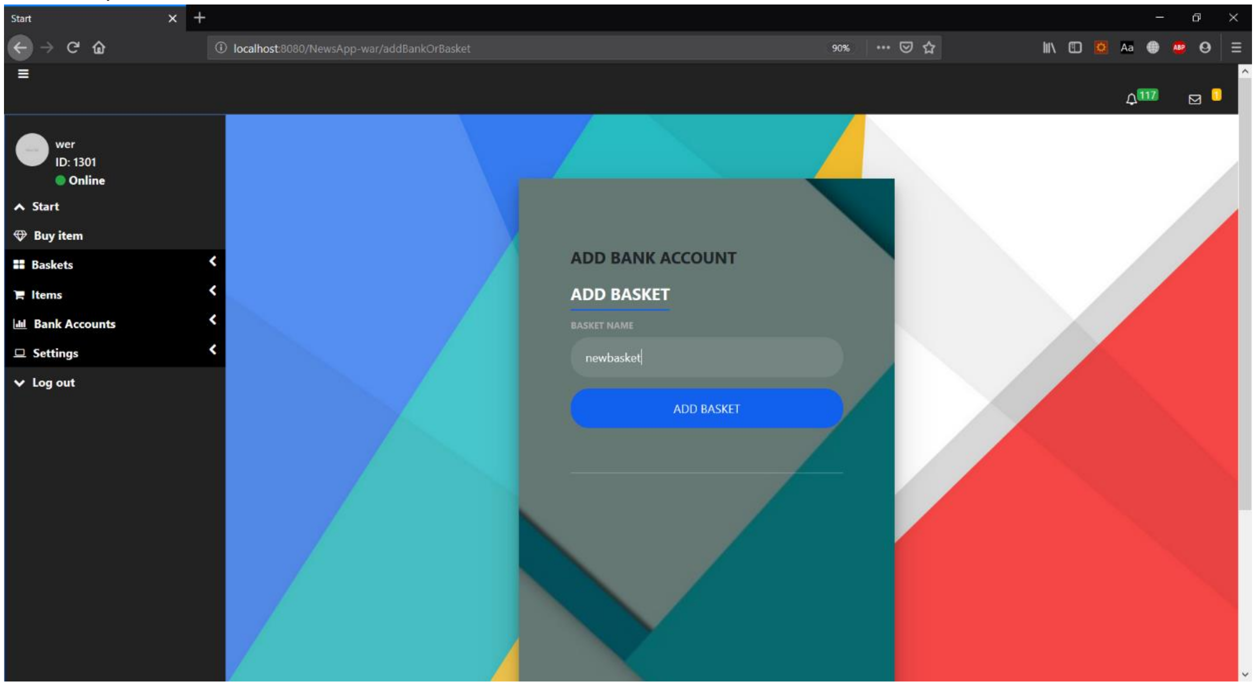


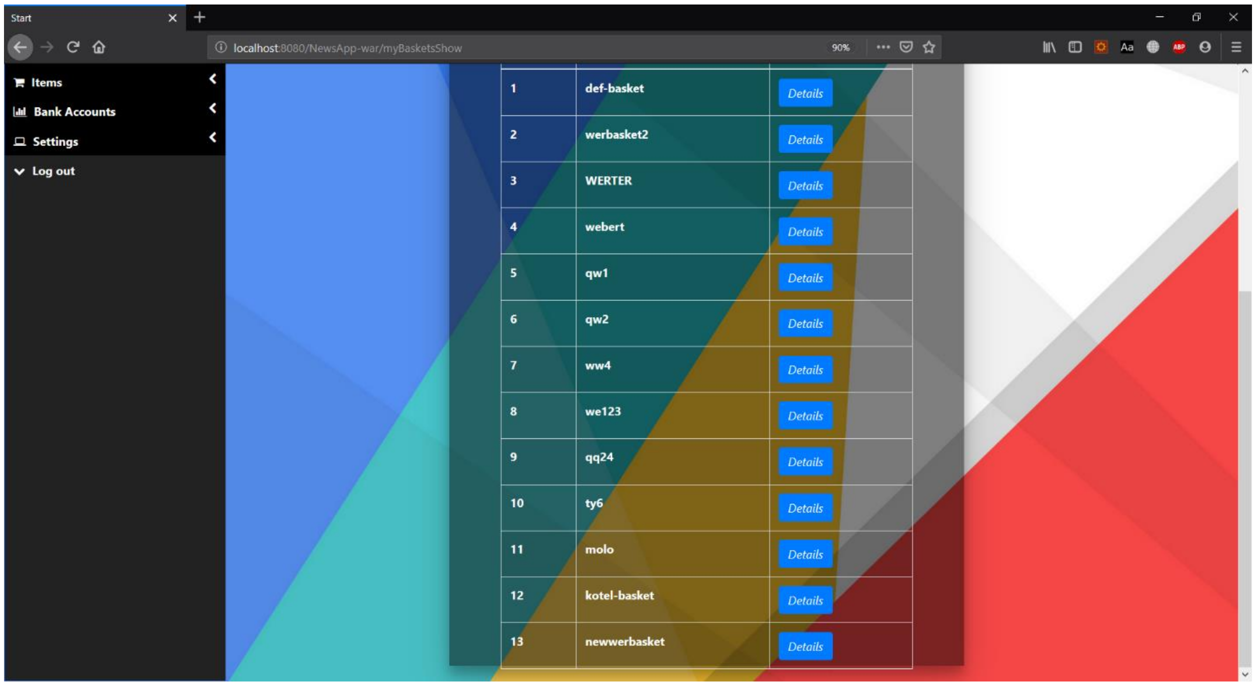


It also allows to show EJB timer status – in right corner with green oval. The timer is the session timer, which log out user if he/she would use page for too long (it's similar mechanism which is used on online banking pages). We can set up time parameter for timer here, but it's set for 300sec. by default. With every page reload we can see how many sec. we have until logout happens. By clicking on a bell we can expand info-windows and check the details. Timer starts when the server starts. Each user can easily log in after logout.



One can check assigned bank account or baskets by clicking in proper button on left menu (screen of previous page). User can also add new account and new basket. Process of creating account works with JS validation (see top screen). Each created new basket is automatically bounded with current user. Info about current user and current basket are stored in ManagementSessionBean – the stateful bean which handle operations with current user and maintain

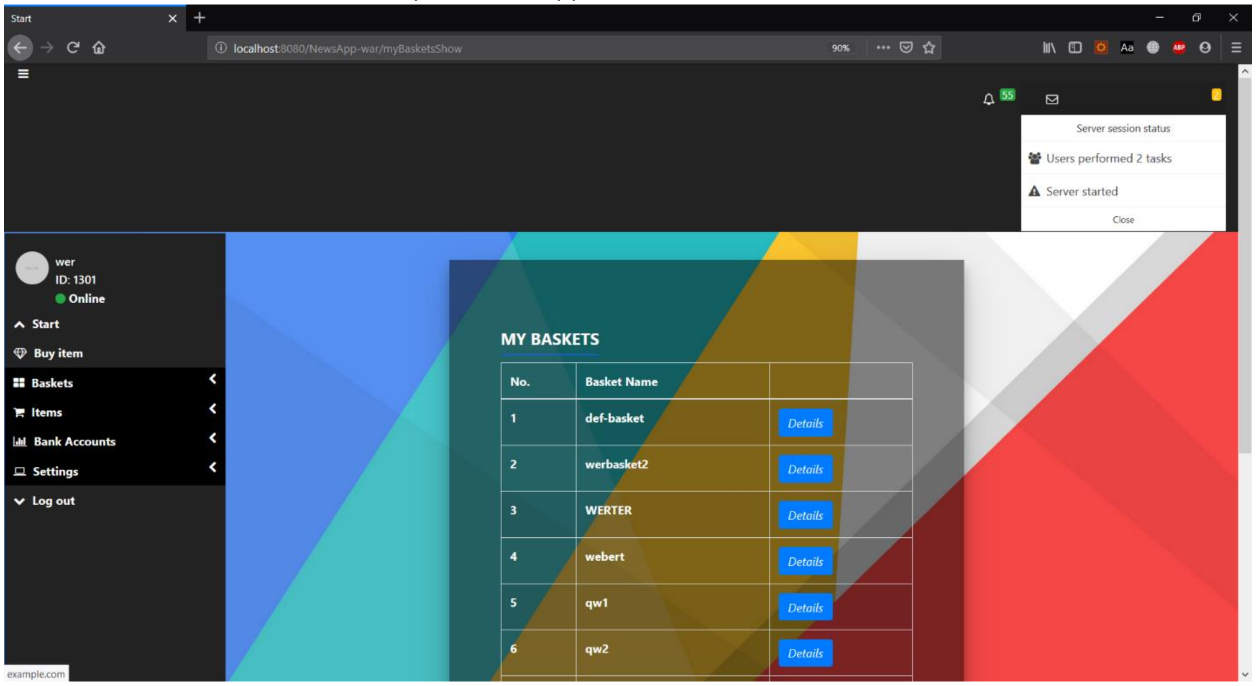




localhost:8080/NewsApp-war/myBasketsShow

1	def-basket	Details
2	werbasket2	Details
3	WERTER	Details
4	webert	Details
5	qw1	Details
6	qw2	Details
7	ww4	Details
8	we123	Details
9	qq24	Details
10	ty6	Details
11	molo	Details
12	kotel-basket	Details
13	newwerbasket	Details

Created basket is seen in My Baskets at the bottom of the list. Each object is created using MessageDrivenBean which transfer information and takes care about the order (messages are in queue). Using persitance manager object (and entities) are created. Check code for more details. MDB is wrapped by Interceptor which catches every change and execution of methods in MDB. When some operation occurs, Interceptor noticed it, send information to SingletonBean which stores info about actions handled by all users during whole server session. Then, informations are displayed in top right corner (yellow oval). User can check how ‘heavy’ or ‘busy’ is system in particular moment. SingletonBean takes also care of creation initial number of Items and Basket, when the server starts. So when you sign in first time and create some object (as bank account or basket) – initial items and basket would be generated for your use. Moreover, each basket has details-button, which uses REST services to get data about particular basket (the same with bank accounts). Parameter in REST request is id of basket. Data are presented in basic XML, since it’s not the crucial part of the application.



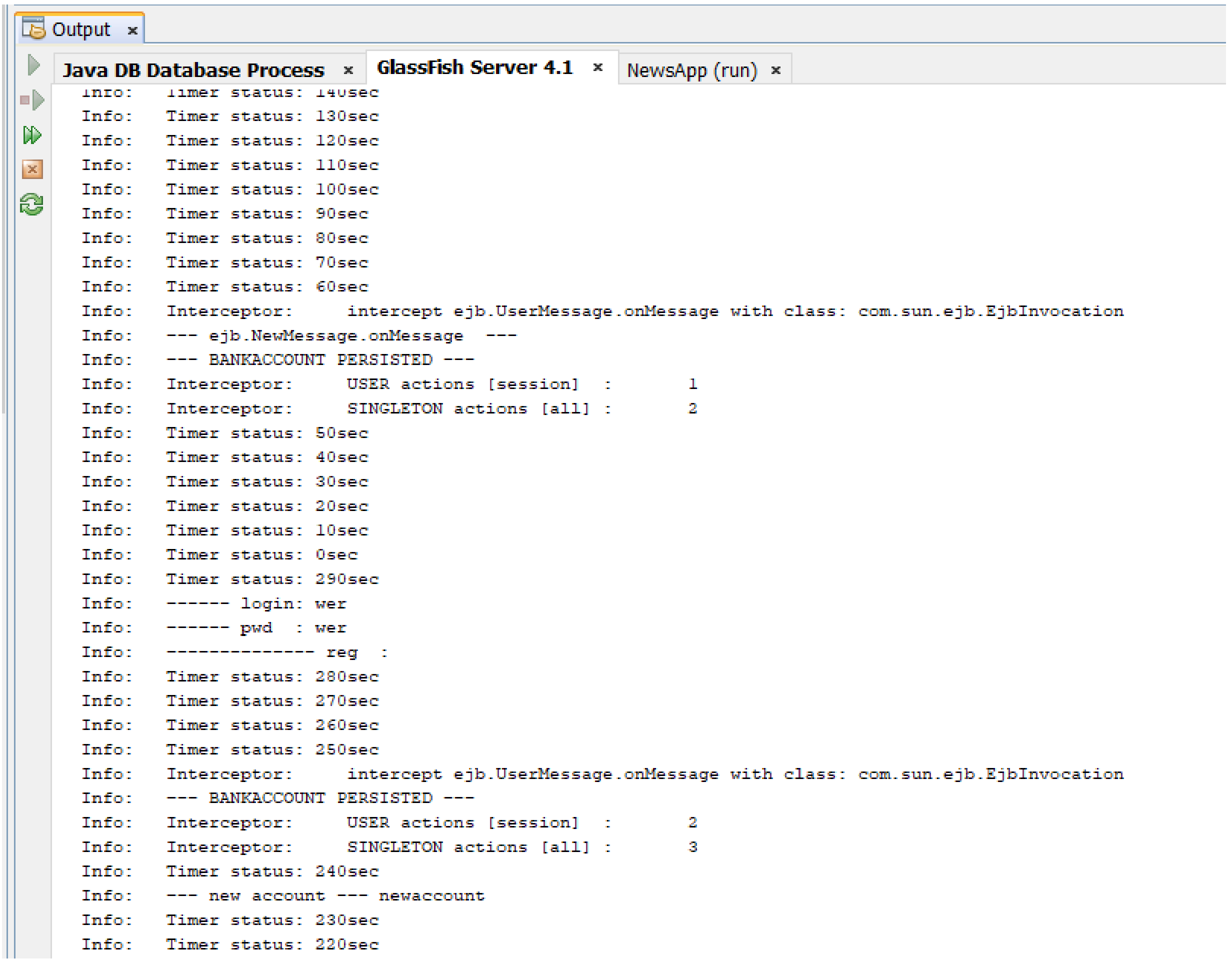
localhost:8080/NewsApp-war/myBasketsShow

Server session status
Users performed 2 tasks
Server started
Close

wer
ID: 1301
Online

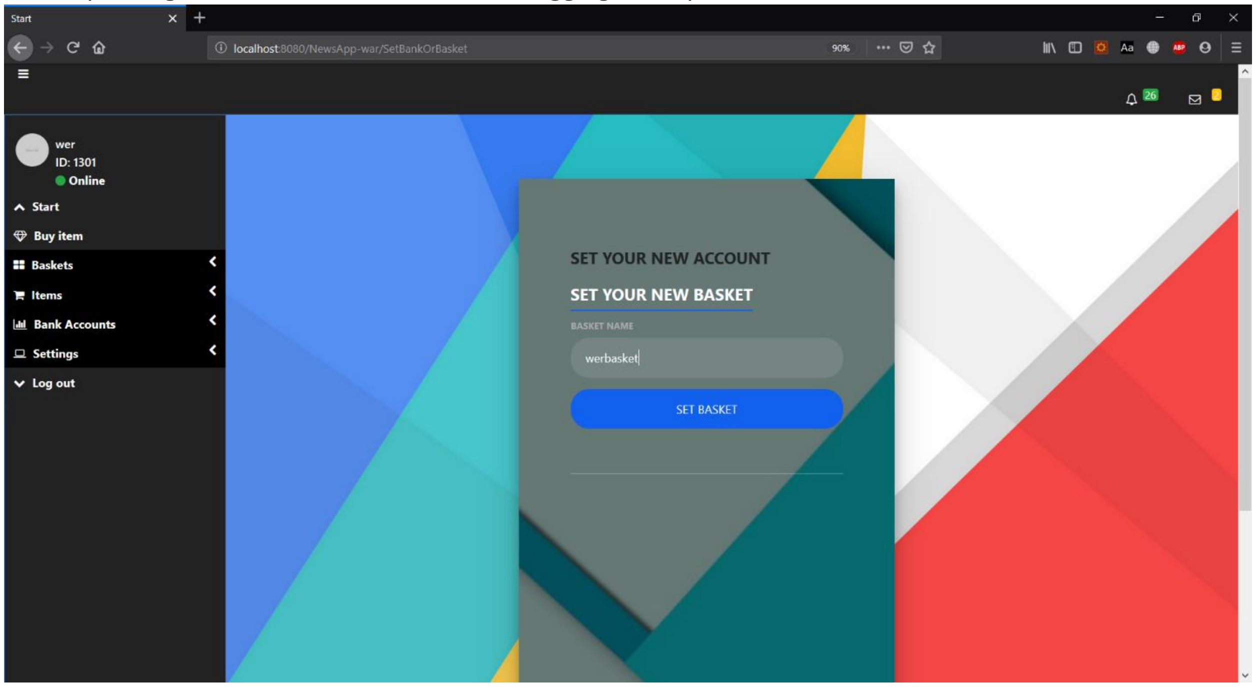
MY BASKETS

No.	Basket Name	
1	def-basket	Details
2	werbasket2	Details
3	WERTER	Details
4	webert	Details
5	qw1	Details
6	qw2	Details

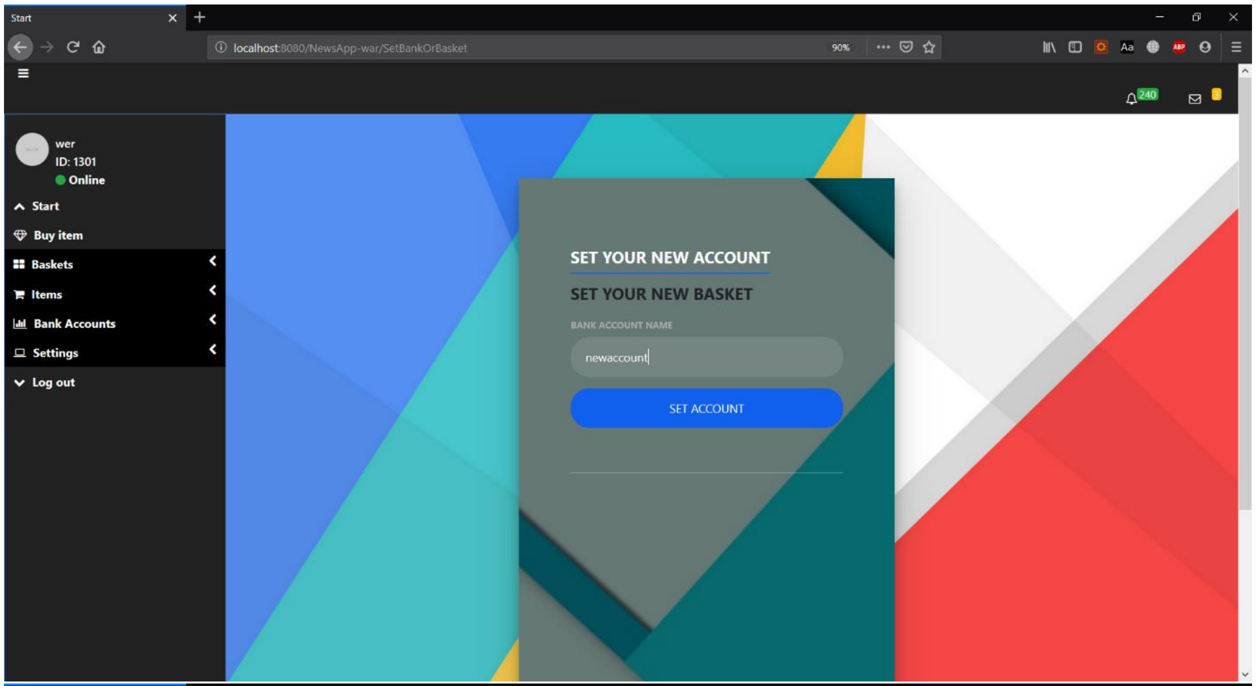


```
Info:   Timer status: 140sec
Info:   Timer status: 130sec
Info:   Timer status: 120sec
Info:   Timer status: 110sec
Info:   Timer status: 100sec
Info:   Timer status: 90sec
Info:   Timer status: 80sec
Info:   Timer status: 70sec
Info:   Timer status: 60sec
Info:   Interceptor:     intercept ejb.UserMessage.onMessage with class: com.sun.ejb.EjbInvocation
Info:   --- ejb.NewMessage.onMessage  ---
Info:   --- BANKACCOUNT PERSISTED ---
Info:   Interceptor:     USER actions [session]  :       1
Info:   Interceptor:     SINGLETON actions [all] :       2
Info:   Timer status: 50sec
Info:   Timer status: 40sec
Info:   Timer status: 30sec
Info:   Timer status: 20sec
Info:   Timer status: 10sec
Info:   Timer status: 0sec
Info:   Timer status: 290sec
Info:   ------ login: wer
Info:   ------ pwd  : wer
Info:   -------------- reg  :
Info:   Timer status: 280sec
Info:   Timer status: 270sec
Info:   Timer status: 260sec
Info:   Timer status: 250sec
Info:   Interceptor:     intercept ejb.UserMessage.onMessage with class: com.sun.ejb.EjbInvocation
Info:   --- BANKACCOUNT PERSISTED ---
Info:   Interceptor:     USER actions [session]  :       2
Info:   Interceptor:     SINGLETON actions [all] :       3
Info:   Timer status: 240sec
Info:   --- new account --- newaccount
Info:   Timer status: 230sec
Info:   Timer status: 220sec
```

Interceptor logs, Timer status and info about logging attempt [in console]



User can also switch from one basket to another and from one account to different. If basket/bankaccount does not exist – page will reload and nothing happened. If name is wrong – you can see error page with Return button.



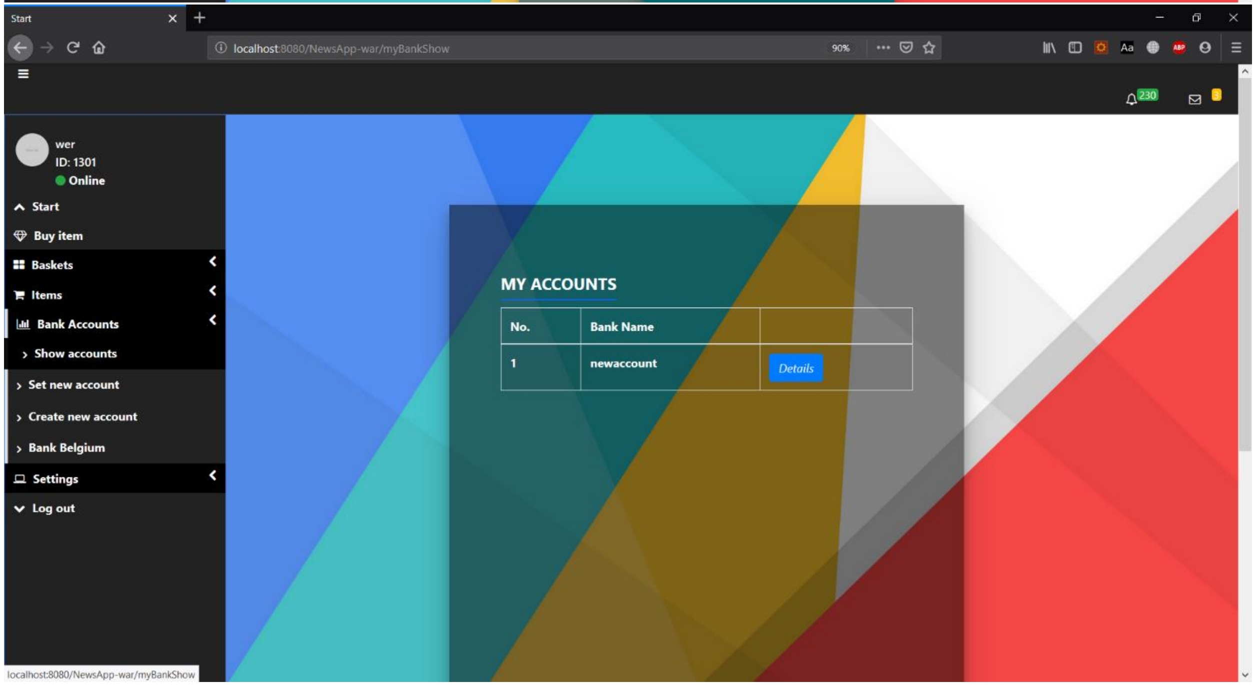
SET YOUR NEW ACCOUNT

SET YOUR NEW BASKET

BANK ACCOUNT NAME

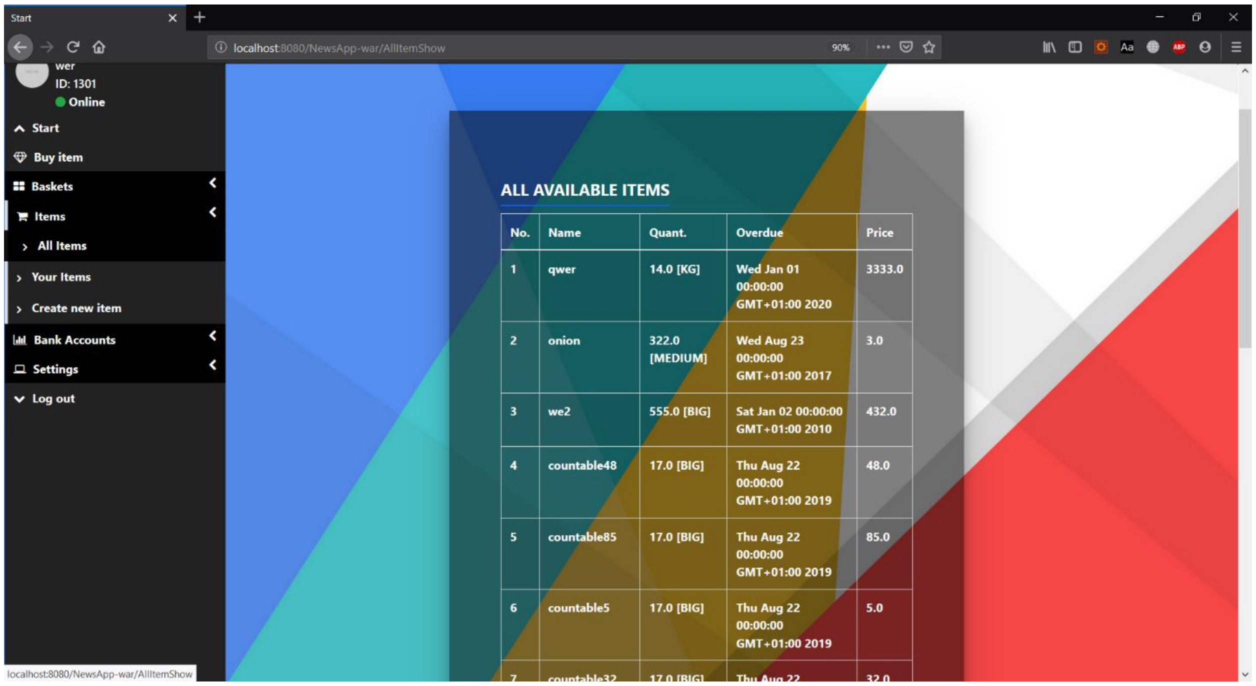
newaccount

SET ACCOUNT



MY ACCOUNTS

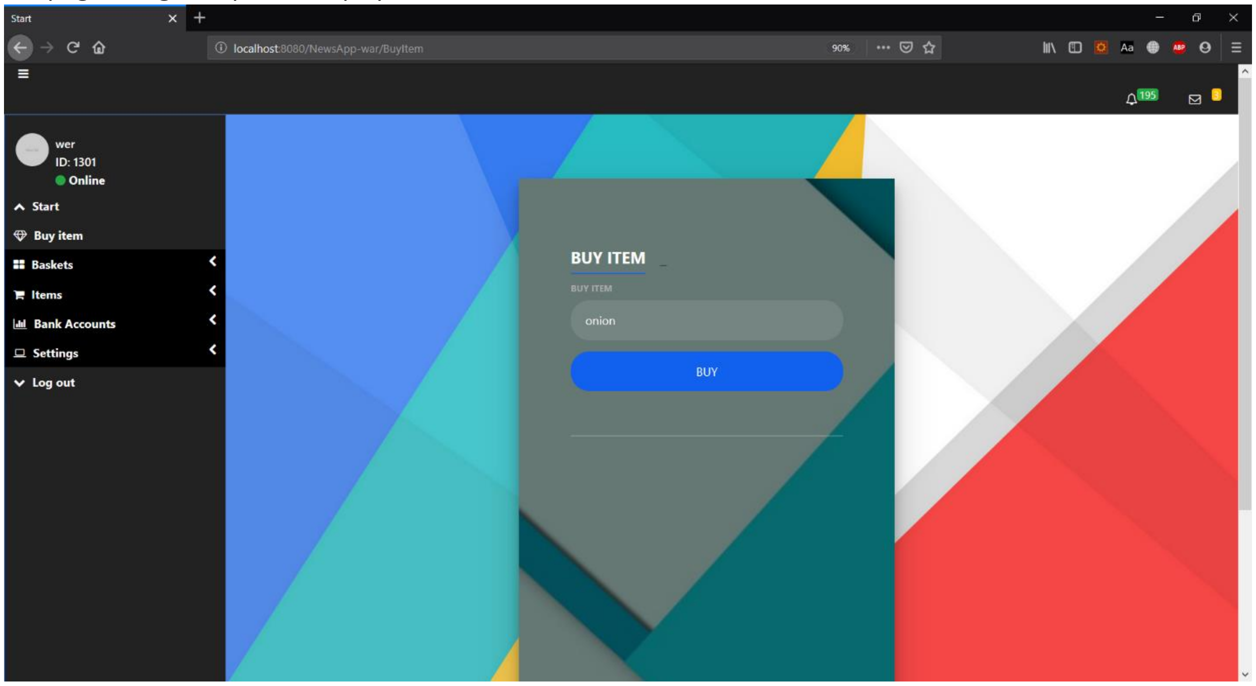
No.	Bank Name	
1	newaccount	Details

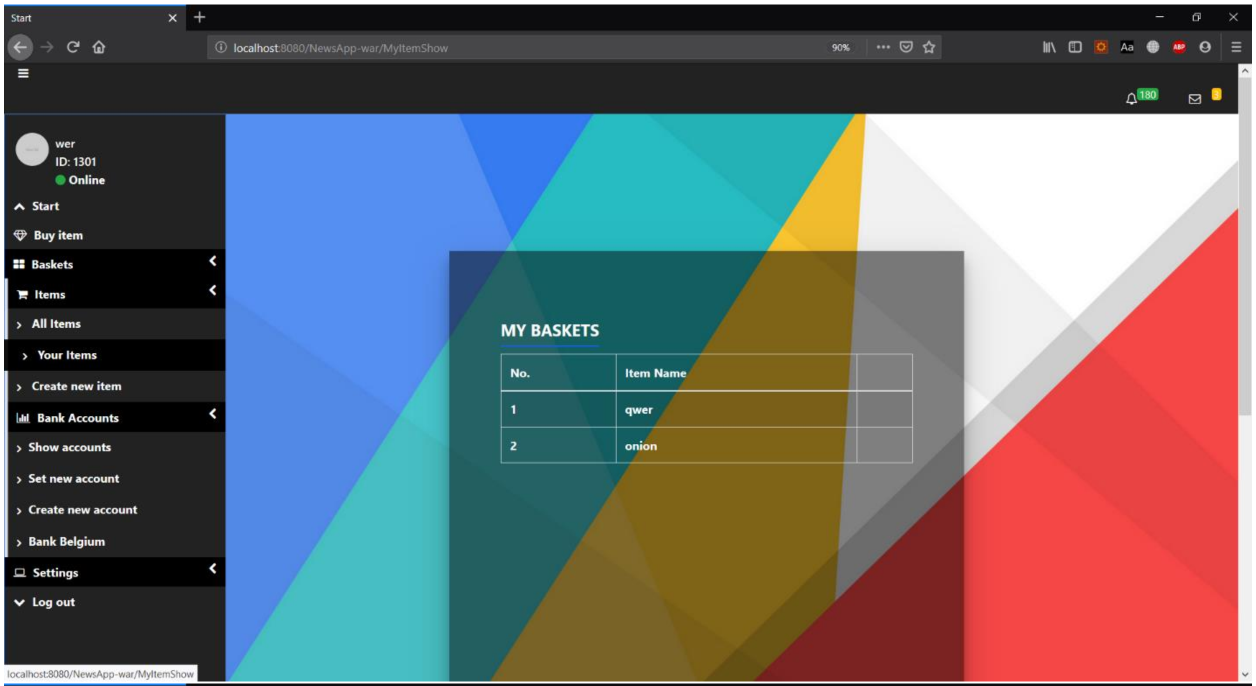


ALL AVAILABLE ITEMS

No.	Name	Quant.	Overdue	Price
1	qwer	14.0 [KG]	Wed Jan 01 00:00:00 GMT+01:00 2020	3333.0
2	onion	322.0 [MEDIUM]	Wed Aug 23 00:00:00 GMT+01:00 2017	3.0
3	we2	555.0 [BIG]	Sat Jan 02 00:00:00 GMT+01:00 2010	432.0
4	countable48	17.0 [BIG]	Thu Aug 22 00:00:00 GMT+01:00 2019	48.0
5	countable85	17.0 [BIG]	Thu Aug 22 00:00:00 GMT+01:00 2019	85.0
6	countable5	17.0 [BIG]	Thu Aug 22 00:00:00 GMT+01:00 2019	5.0
7	countable32	17.0 [BIG]	Thu Aug 22	32.0

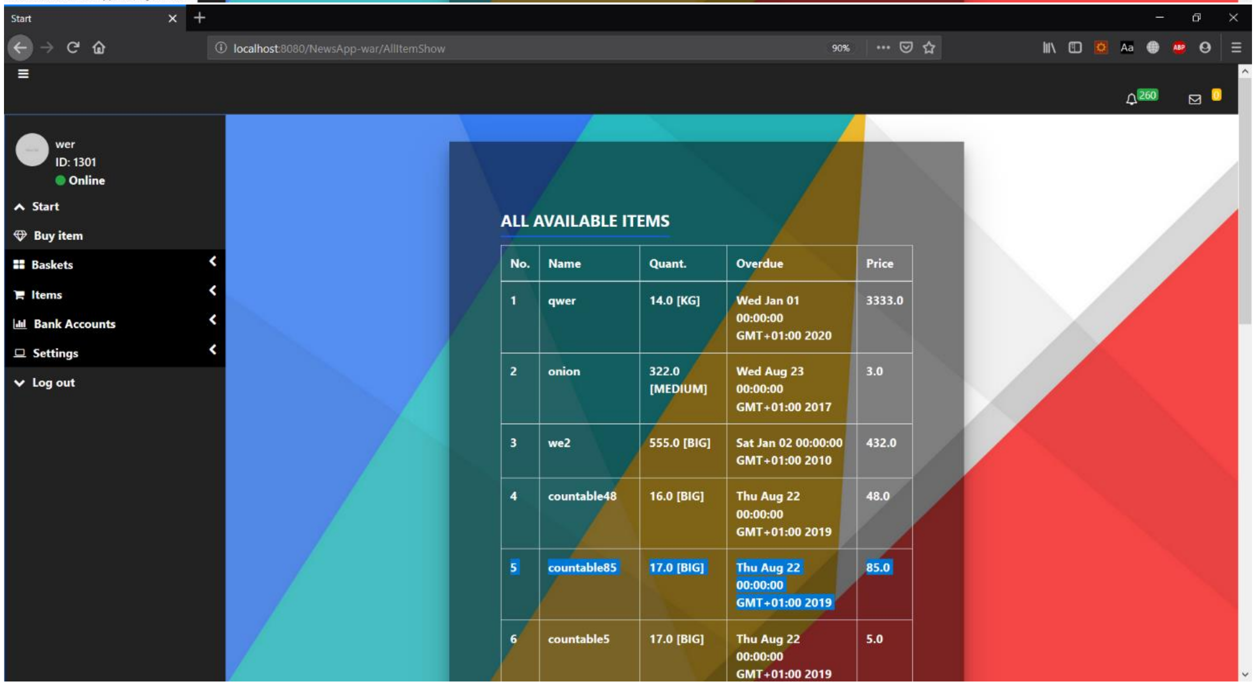
The most important part is ‘buying’/’assinging’ items. We can check item list with basic info about each one. Item can have two types – countable and uncountable which determines their attributes such as price/pricePerWeight or size/unit. Last ones are Enum type. When we buy some item (by inserting item name on buy-item page) the quantity of items on main All-Item list decrements (there are -1 items) and bought item is assigned to current basket and current user. We can check our items on MyItem list – items are retrieved using SQL query in UserFacade method, transformed by Servlet and represented on JSP page using JSP special loop syntax.





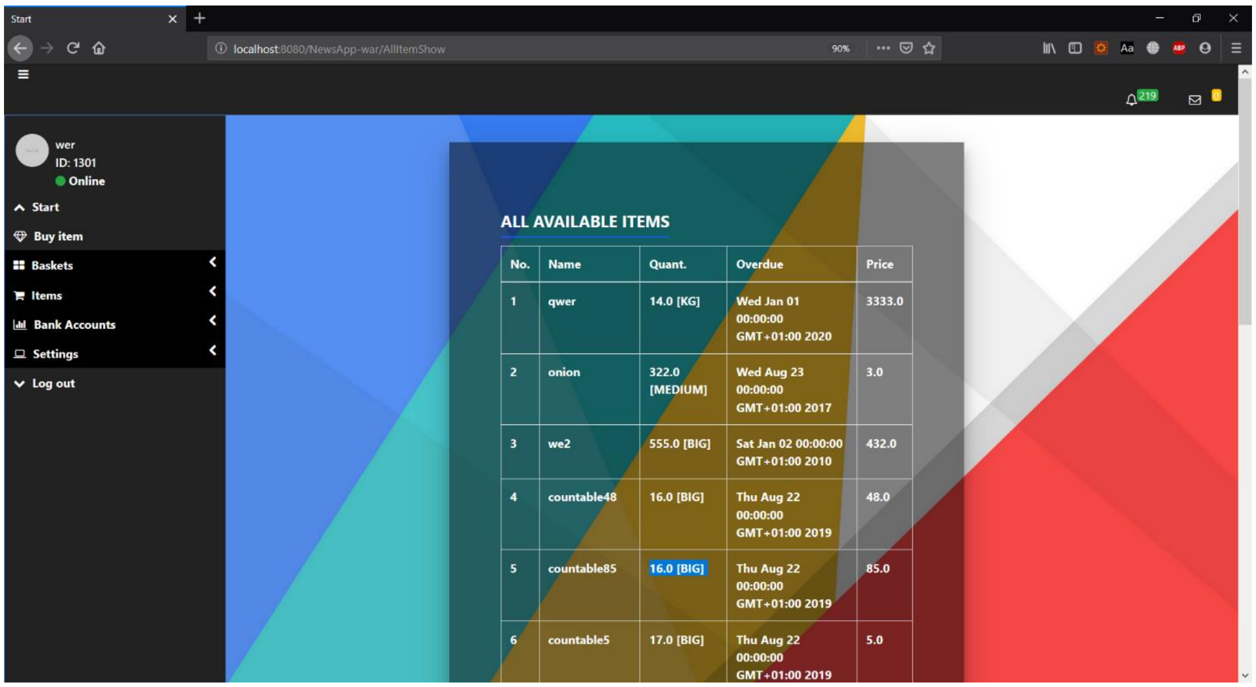
MY BASKETS

No.	Item Name	
1	qwer	
2	onion	



ALL AVAILABLE ITEMS

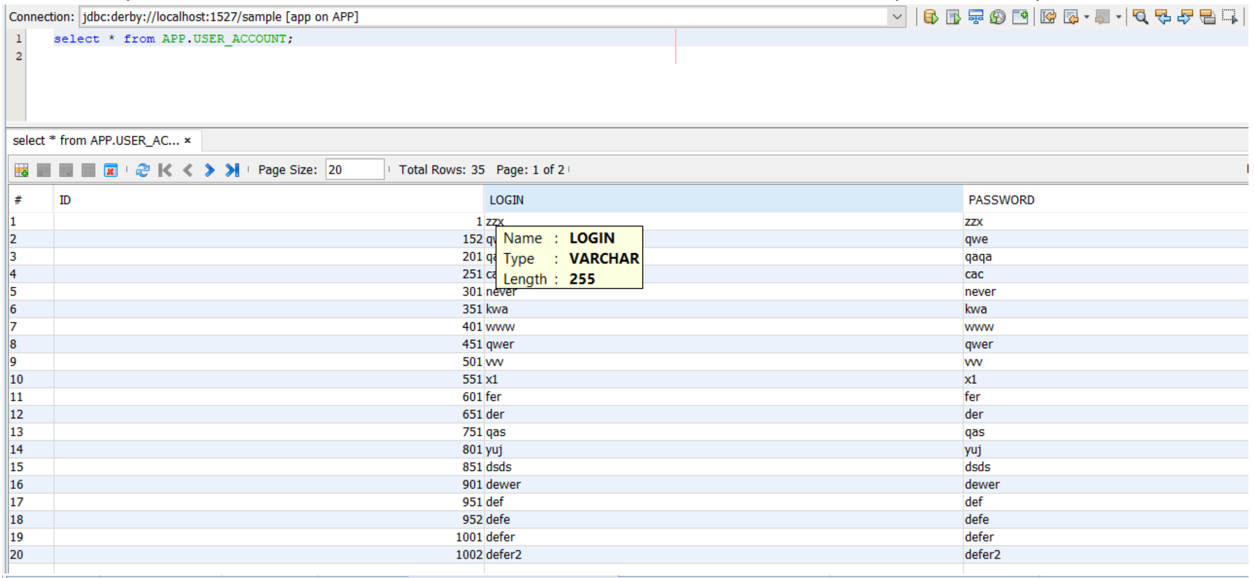
No.	Name	Quant.	Overdue	Price
1	qwer	14.0 [KG]	Wed Jan 01 00:00:00 GMT+01:00 2020	3333.0
2	onion	322.0 [MEDIUM]	Wed Aug 23 00:00:00 GMT+01:00 2017	3.0
3	we2	555.0 [BIG]	Sat Jan 02 00:00:00 GMT+01:00 2010	432.0
4	countable48	16.0 [BIG]	Thu Aug 22 00:00:00 GMT+01:00 2019	48.0
5	countable85	17.0 [BIG]	Thu Aug 22 00:00:00 GMT+01:00 2019	85.0
6	countable5	17.0 [BIG]	Thu Aug 22 00:00:00 GMT+01:00 2019	5.0



ALL AVAILABLE ITEMS

No.	Name	Quant.	Overdue	Price
1	qwer	14.0 [KG]	Wed Jan 01 00:00:00 GMT+01:00 2020	3333.0
2	onion	322.0 [MEDIUM]	Wed Aug 23 00:00:00 GMT+01:00 2017	3.0
3	we2	555.0 [BIG]	Sat Jan 02 00:00:00 GMT+01:00 2010	432.0
4	countable48	16.0 [BIG]	Thu Aug 22 00:00:00 GMT+01:00 2019	48.0
5	countable85	16.0 [BIG]	Thu Aug 22 00:00:00 GMT+01:00 2019	85.0
6	countable5	17.0 [BIG]	Thu Aug 22 00:00:00 GMT+01:00 2019	5.0

Below, there are screenshots from the database. As we can see – tables are connected to each other. For example USER_BANKACCOUNTS contains ID of User and ID of Bank which are in ManyToMany relationship. User-Basket and Basket-Item works similar, but with OneToMany relationship.



Connection: jdbc:derby://localhost:1527/sample [app on APP]

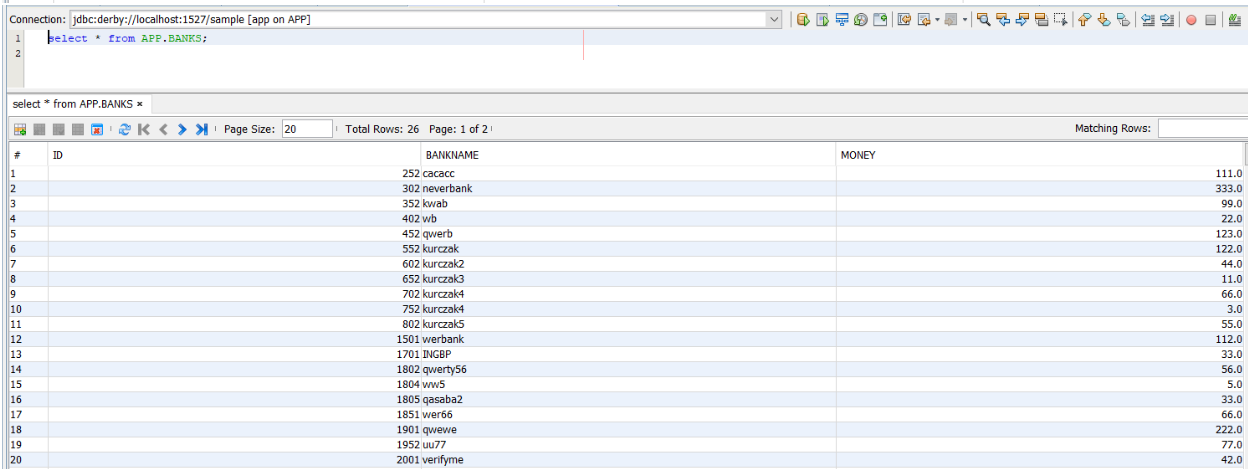
```
select * from APP.USER_ACCOUNT;
```

select * from APP.USER_AC... ×

Page Size: 20 Total Rows: 35 Page: 1 of 2

Name : LOGIN
Type : VARCHAR
Length : 255

#	ID	LOGIN	PASSWORD
1	1	zzx	zzx
2	152	q	qwe
3	201	qa	qaqa
4	251	ca	cac
5	301	never	never
6	351	kwa	kwa
7	401	www	www
8	451	qwer	qwer
9	501	vv	vv
10	551	x1	x1
11	601	fer	fer
12	651	der	der
13	751	qas	qas
14	801	yuj	yuj
15	851	dsds	dsds
16	901	dewer	dewer
17	951	def	def
18	952	defe	defe
19	1001	defer	defer
20	1002	defer2	defer2



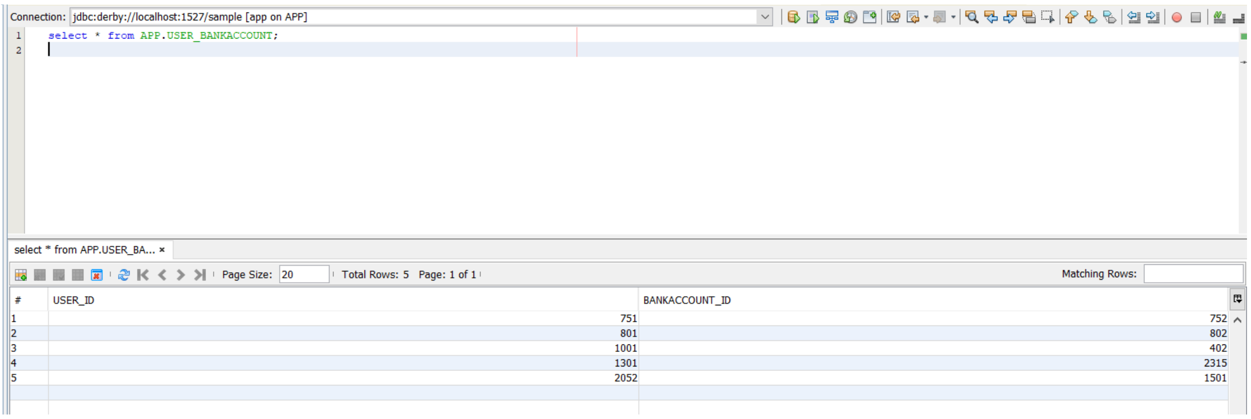
Connection: jdbc:derby://localhost:1527/sample [app on APP]

```
select * from APP.BANKS;
```

select * from APP.BANKS ×

Page Size: 20 Total Rows: 26 Page: 1 of 2 Matching Rows:

#	ID	BANKNAME	MONEY
1	252	cacacc	111.0
2	302	neverbank	333.0
3	352	kwab	99.0
4	402	wb	22.0
5	452	qwerb	123.0
6	552	kurczak	122.0
7	602	kurczak2	44.0
8	652	kurczak3	11.0
9	702	kurczak4	66.0
10	752	kurczak4	3.0
11	802	kurczak5	55.0
12	1501	werbank	112.0
13	1701	INGBP	33.0
14	1802	qwerty56	56.0
15	1804	ww5	5.0
16	1805	qasaba2	33.0
17	1851	wer66	66.0
18	1901	qwewe	222.0
19	1952	uu77	77.0
20	2001	verifyme	42.0



Connection: jdbc:derby://localhost:1527/sample [app on APP]

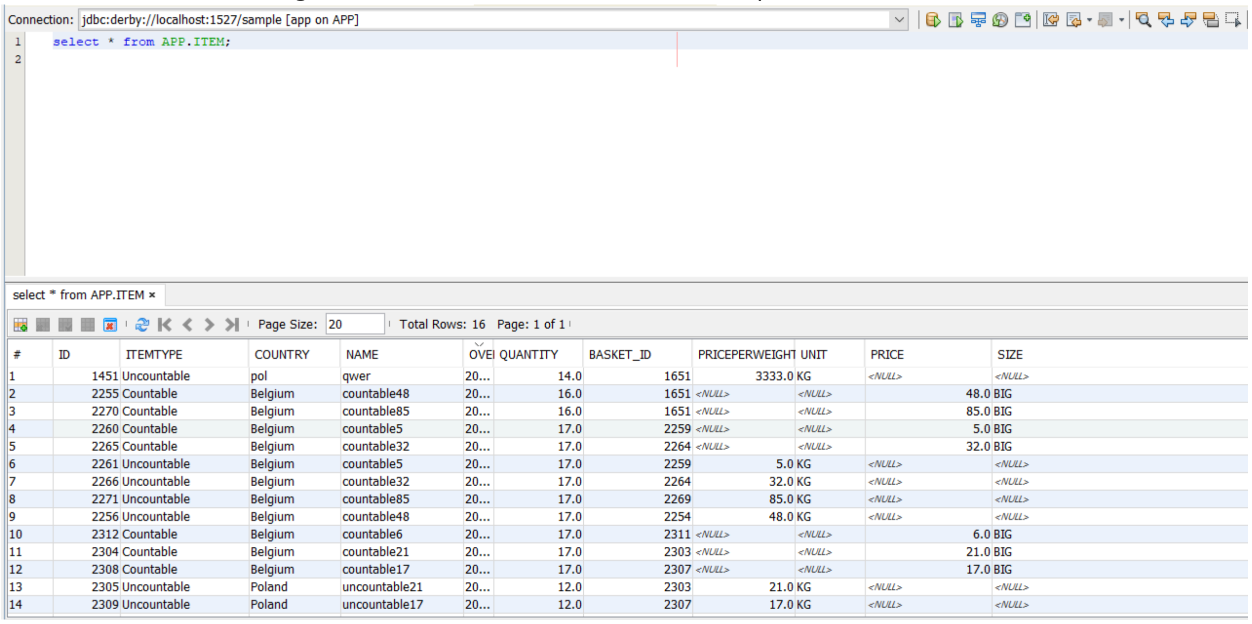
```
select * from APP.USER_BANKACCOUNT;
```

select * from APP.USER_BA... ×

Page Size: 20 Total Rows: 5 Page: 1 of 1 Matching Rows:

#	USER_ID	BANKACCOUNT_ID
1	751	752
2	801	802
3	1001	402
4	1301	2315
5	2052	1501

Item database table contains information about both types of Items (Countable and Uncountable), to make process of querying easier and faster. ITEMTYPE attribute defines type of Item. Since Countable has no Unit and PrecePerWeight attributes – these fields are nulled by default.



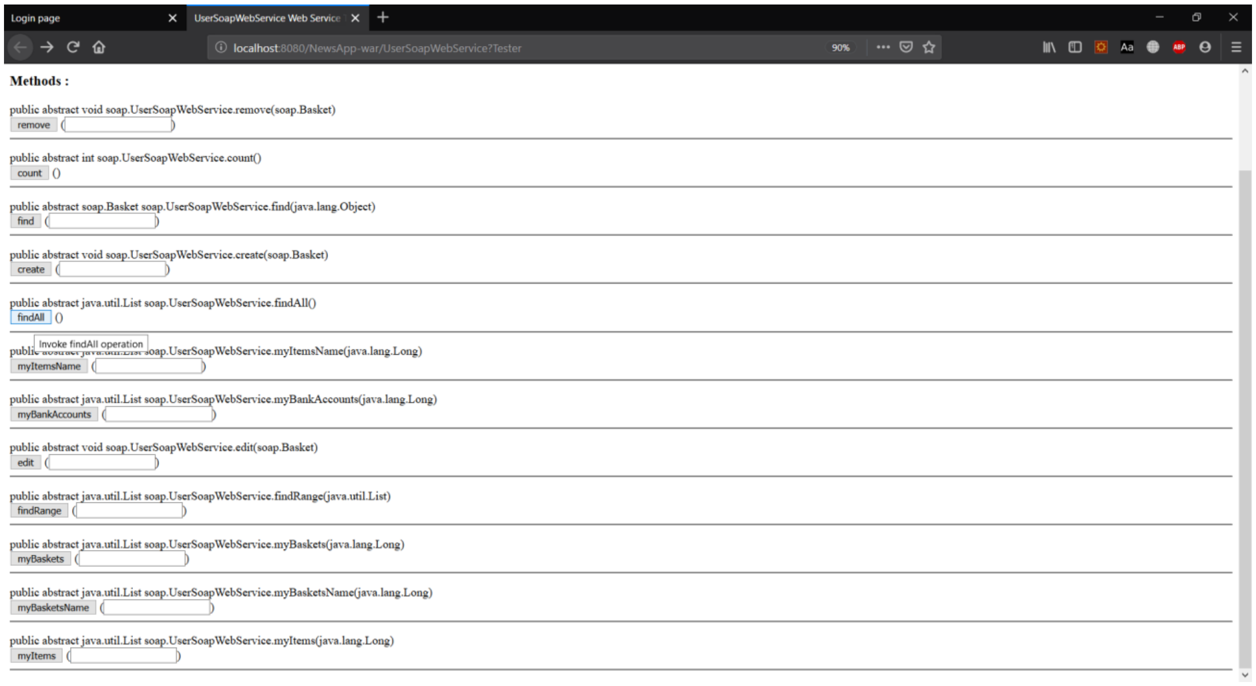
Connection: jdbc:derby://localhost:1527/sample [app on APP]

```
select * from APP.ITEM;
```

select * from APP.ITEM ×

Page Size: 20 Total Rows: 16 Page: 1 of 1

#	ID	ITEMTYPE	COUNTRY	NAME	OVEI	QUANTITY	BASKET_ID	PRICEPERWEIGHT	UNIT	PRICE	SIZE
1	1451	Uncountable	pol	qwer	20...	14.0	1651	3333.0	KG	<NULL>	<NULL>
2	2255	Countable	Belgium	countable48	20...	16.0	1651	<NULL>	<NULL>	48.0	BIG
3	2270	Countable	Belgium	countable85	20...	16.0	1651	<NULL>	<NULL>	85.0	BIG
4	2260	Countable	Belgium	countable5	20...	17.0	2259	<NULL>	<NULL>	5.0	BIG
5	2265	Countable	Belgium	countable32	20...	17.0	2264	<NULL>	<NULL>	32.0	BIG
6	2261	Uncountable	Belgium	countable5	20...	17.0	2259	5.0	KG	<NULL>	<NULL>
7	2266	Uncountable	Belgium	countable32	20...	17.0	2264	32.0	KG	<NULL>	<NULL>
8	2271	Uncountable	Belgium	countable85	20...	17.0	2269	85.0	KG	<NULL>	<NULL>
9	2256	Uncountable	Belgium	countable48	20...	17.0	2254	48.0	KG	<NULL>	<NULL>
10	2312	Countable	Belgium	countable6	20...	17.0	2311	<NULL>	<NULL>	6.0	BIG
11	2304	Countable	Belgium	countable21	20...	17.0	2303	<NULL>	<NULL>	21.0	BIG
12	2308	Countable	Belgium	countable17	20...	17.0	2307	<NULL>	<NULL>	17.0	BIG
13	2305	Uncountable	Poland	uncountable21	20...	12.0	2303	21.0	KG	<NULL>	<NULL>
14	2309	Uncountable	Poland	uncountable17	20...	12.0	2307	17.0	KG	<NULL>	<NULL>



Login page | UserSoapWebService Web Service

localhost:8080/NewsApp-war/UserSoapWebService?Tester

Methods :

public abstract void soap.UserSoapWebService.remove(soap.Basket)
remove ( )

public abstract int soap.UserSoapWebService.count()
count ()

public abstract soap.Basket soap.UserSoapWebService.find(java.lang.Object)
find ( )

public abstract void soap.UserSoapWebService.create(soap.Basket)
create ( )

public abstract java.util.List soap.UserSoapWebService.findAll()
findAll ()

Invoke findAll operation

public abstract java.util.List soap.UserSoapWebService.myItemsName(java.lang.Long)
myItemsName ( )

public abstract java.util.List soap.UserSoapWebService.myBankAccounts(java.lang.Long)
myBankAccounts ( )

public abstract void soap.UserSoapWebService.edit(soap.Basket)
edit ( )

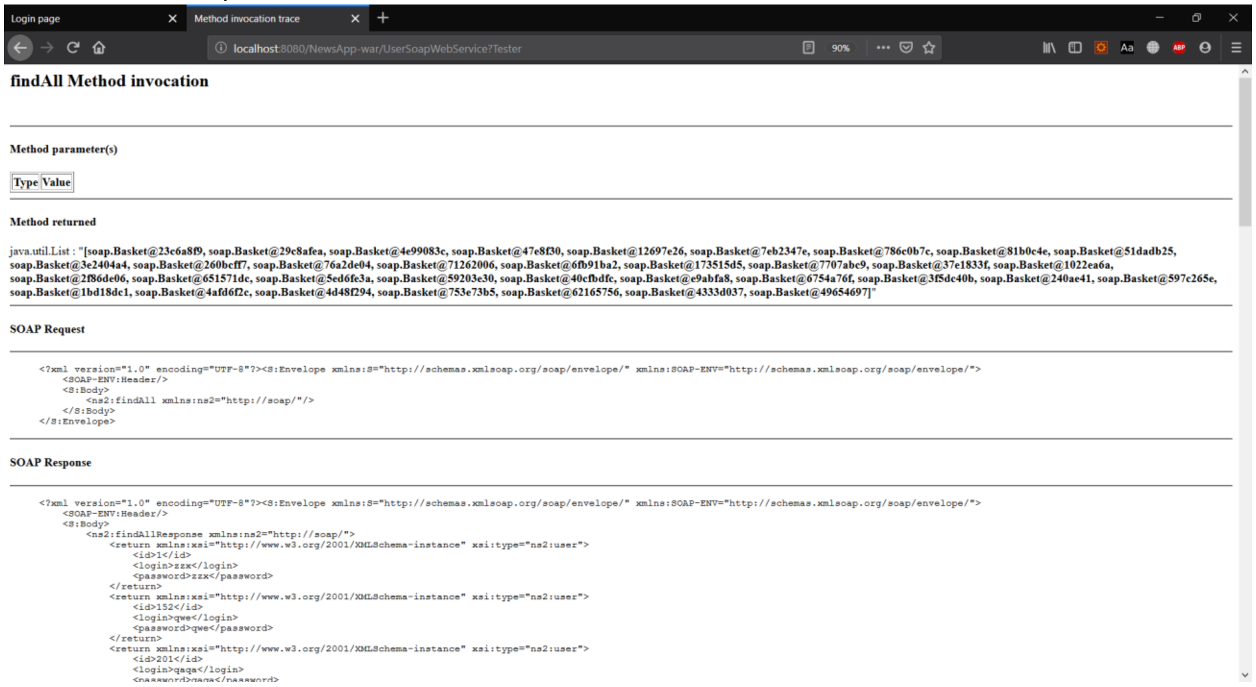
public abstract java.util.List soap.UserSoapWebService.findRange(java.util.List)
findRange ( )

public abstract java.util.List soap.UserSoapWebService.myBaskets(java.lang.Long)
myBaskets ( )

public abstract java.util.List soap.UserSoapWebService.myBasketsName(java.lang.Long)
myBasketsName ( )

public abstract java.util.List soap.UserSoapWebService.myItems(java.lang.Long)
myItems ( )

SOAP Client in the application allows us to modify almost every part of an app. We can easily create/edit/remove each entity, as well as make SOAP requests with various parameters. Since the core of an app is based on Servlets and REST services – SOAP web services and clients were added in addition and are not developed on the same level as Servlet/REST services.



Login page | Method invocation trace

localhost:8080/NewsApp-war/UserSoapWebService?Tester

findAll Method invocation

Method parameter(s)

Type	Value

Method returned

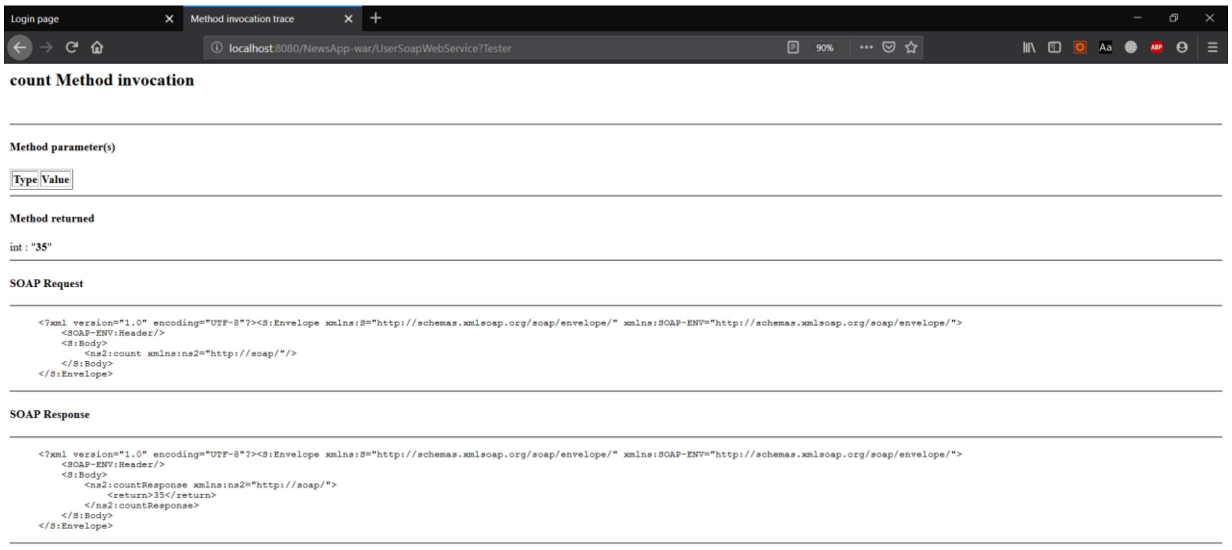
java.util.List : "**[soap.Basket@23c6a8f9, soap.Basket@29c8afea, soap.Basket@4e99083c, soap.Basket@47e8f30, soap.Basket@12697e26, soap.Basket@7eb2347e, soap.Basket@786c0b7c, soap.Basket@81b0c4e, soap.Basket@51dadb25, soap.Basket@3e2404a4, soap.Basket@260bcff7, soap.Basket@76a2de04, soap.Basket@71262006, soap.Basket@6fb91ba2, soap.Basket@173515d5, soap.Basket@7707abc9, soap.Basket@37e1833f, soap.Basket@1022ea6a, soap.Basket@2f86de06, soap.Basket@651571dc, soap.Basket@5ed6fe3a, soap.Basket@59203e30, soap.Basket@40cfbdfc, soap.Basket@e9abfa8, soap.Basket@6754a76f, soap.Basket@3f5dc40b, soap.Basket@240ae41, soap.Basket@597c265e, soap.Basket@1bd18dc1, soap.Basket@4afd6f2c, soap.Basket@4d48f294, soap.Basket@753e73b5, soap.Basket@62165756, soap.Basket@4333d037, soap.Basket@49654697]**"

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Header/>
    <S:Body>
        <ns2:findAll xmlns:ns2="http://soap/"/>
    </S:Body>
</S:Envelope>
```

SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Header/>
    <S:Body>
        <ns2:findAllResponse xmlns:ns2="http://soap/">
            <return xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="ns2:user">
                <id>1</id>
                <login>zzx</login>
                <password>zzx</password>
            </return>
            <return xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="ns2:user">
                <id>152</id>
                <login>qwe</login>
                <password>qwe</password>
            </return>
            <return xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="ns2:user">
                <id>201</id>
                <login>qaqa</login>
                <password>qaqa</password>
```



count Method invocation

Method parameter(s)

Type	Value

Method returned

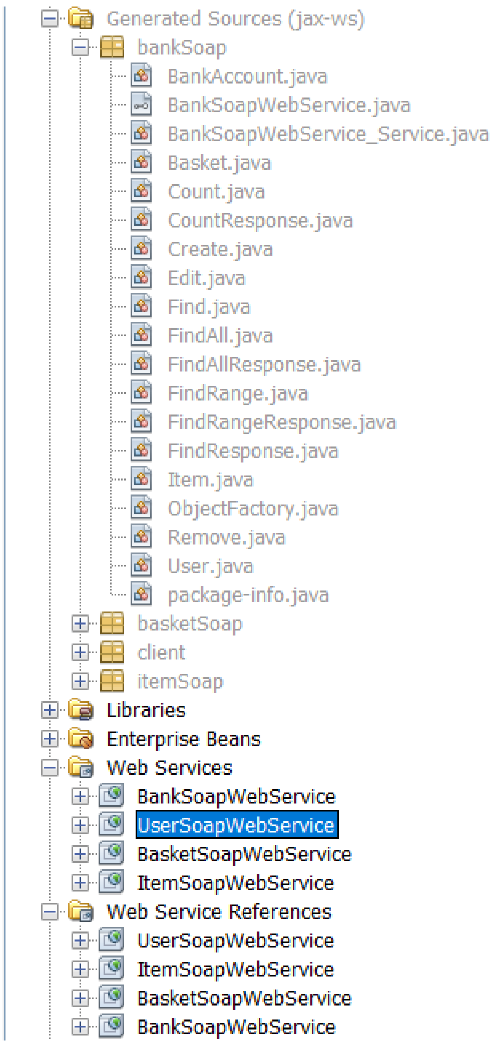
int : "35"

SOAP Request

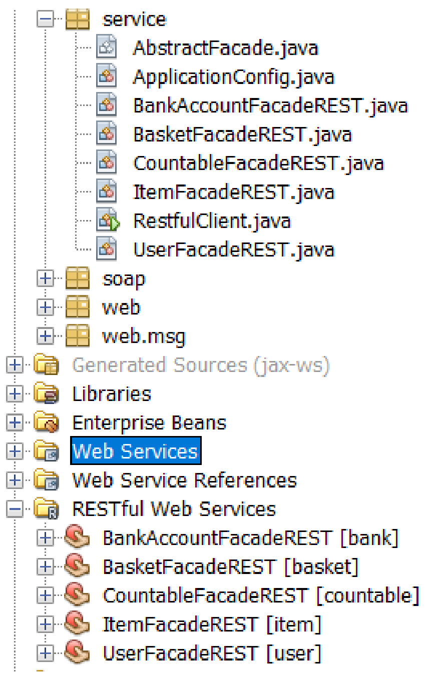
```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Header/>
    <S:Body>
        <ns2:count xmlns:ns2="http://soap/"/>
    </S:Body>
</S:Envelope>
```

SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Header/>
    <S:Body>
        <ns2:countResponse xmlns:ns2="http://soap/">
            <return>35</return>
        </ns2:countResponse>
    </S:Body>
</S:Envelope>
```

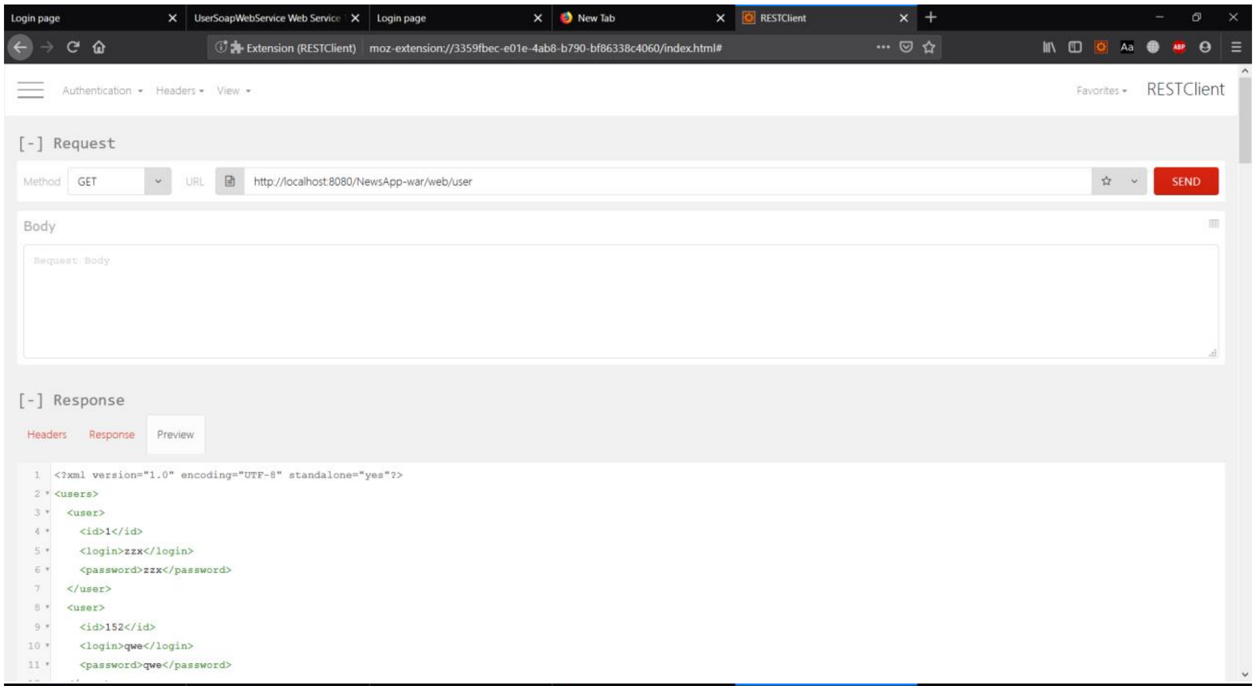


```
Generated Sources (jax-ws)
    bankSoap
        BankAccount.java
        BankSoapWebService.java
        BankSoapWebService_Service.java
        Basket.java
        Count.java
        CountResponse.java
        Create.java
        Edit.java
        Find.java
        FindAll.java
        FindAllResponse.java
        FindRange.java
        FindRangeResponse.java
        FindResponse.java
        Item.java
        ObjectFactory.java
        Remove.java
        User.java
        package-info.java
    basketSoap
    client
    itemSoap
Libraries
Enterprise Beans
Web Services
    BankSoapWebService
    UserSoapWebService
    BasketSoapWebService
    ItemSoapWebService
Web Service References
    UserSoapWebService
    ItemSoapWebService
    BasketSoapWebService
    BankSoapWebService
```



```
    service
        AbstractFacade.java
        ApplicationConfig.java
        BankAccountFacadeREST.java
        BasketFacadeREST.java
        CountableFacadeREST.java
        ItemFacadeREST.java
        RestfulClient.java
        UserFacadeREST.java
    soap
    web
    web.msg
Generated Sources (jax-ws)
Libraries
Enterprise Beans
Web Services
Web Service References
RESTful Web Services
    BankAccountFacadeREST [bank]
    BasketFacadeREST [basket]
    CountableFacadeREST [countable]
    ItemFacadeREST [item]
    UserFacadeREST [user]
```

Besides the fact, that user can make REST requests in app using special buttons, it’s also available to use additional REST client (RestfulClient.java which could also be executed separately) and more advanced REST requests – as presented in screenshots below.


Login page
UserSoapWebService Web Service
Login page
New Tab
RESTClient
Extension (RESTClient) moz-extension://3359fbec-e01e-4ab8-b790-bf86338c4060/index.html#
Authentication
Headers
View
Favorites
RESTClient
Headers
Response
Preview

[-] Request

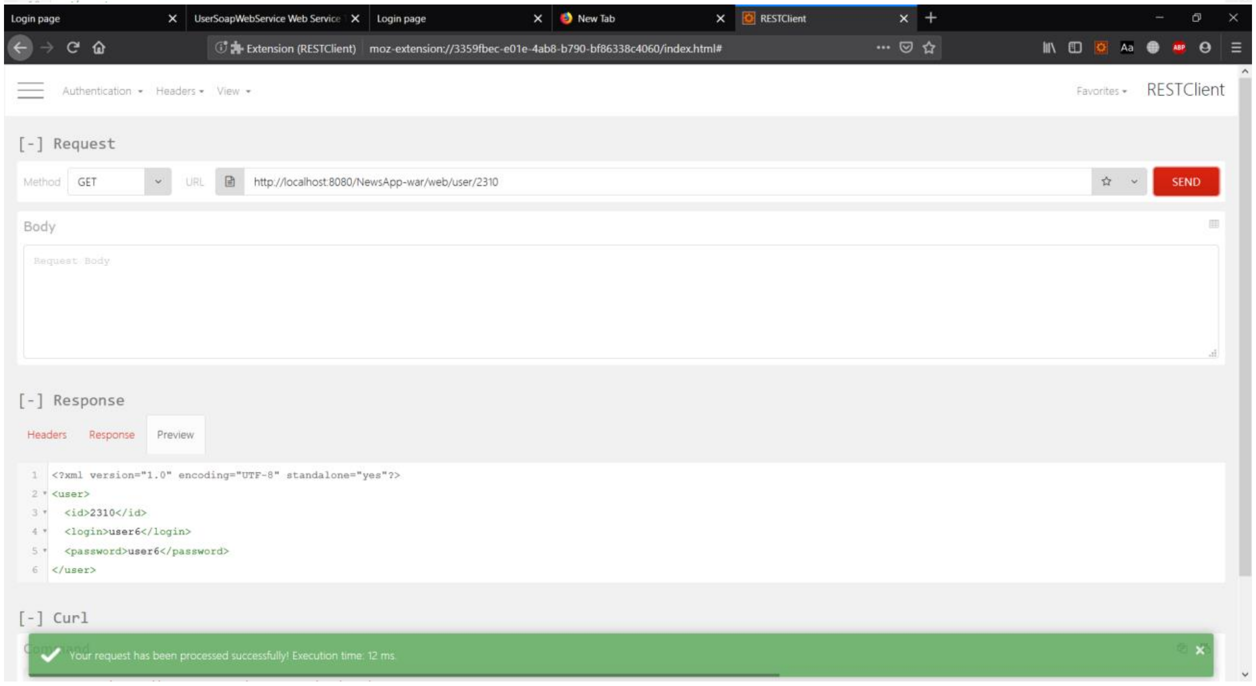
Method GET URL http://localhost:8080/NewsApp-war/web/user SEND

Body

Request Body

[-] Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<users>
  <user>
    <id>1</id>
    <login>zzx</login>
    <password>zzx</password>
  </user>
  <user>
    <id>152</id>
    <login>qwe</login>
    <password>qwe</password>
```


Login page
UserSoapWebService Web Service
Login page
New Tab
RESTClient
Extension (RESTClient) moz-extension://3359fbec-e01e-4ab8-b790-bf86338c4060/index.html#
Authentication
Headers
View
Favorites
RESTClient
Headers
Response
Preview

[-] Request

Method GET URL http://localhost:8080/NewsApp-war/web/user/2310 SEND

Body

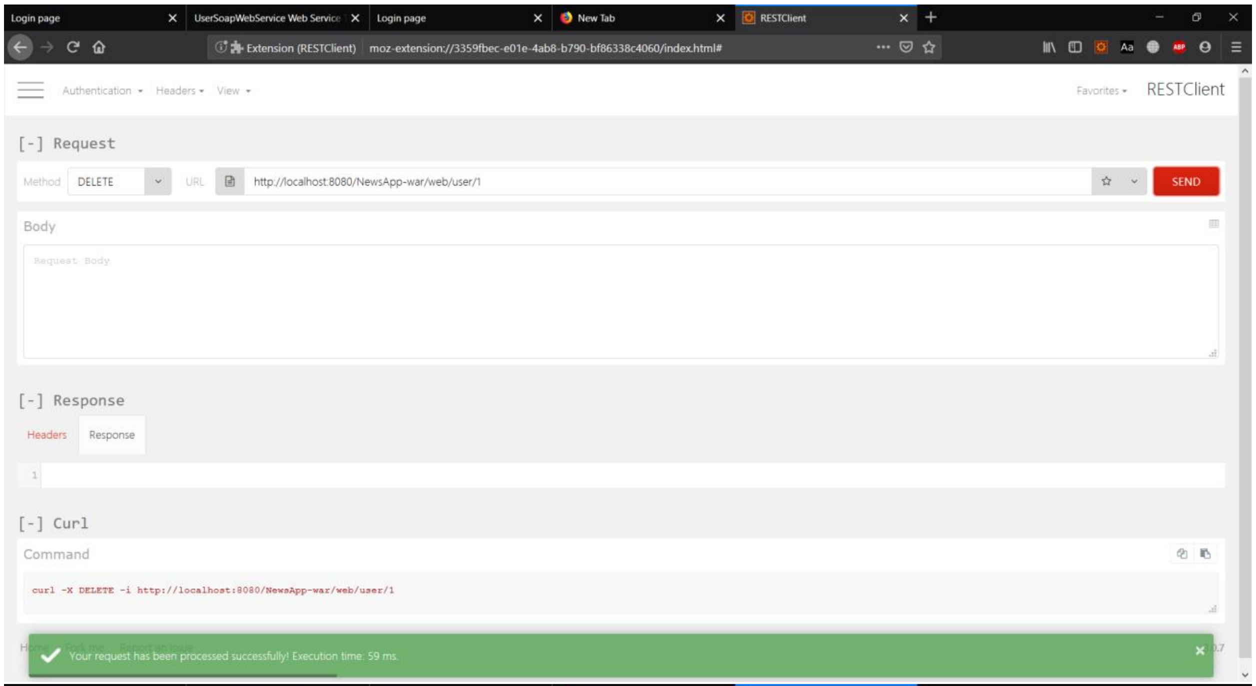
Request Body

[-] Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<user>
  <id>2310</id>
  <login>user6</login>
  <password>user6</password>
</user>
```

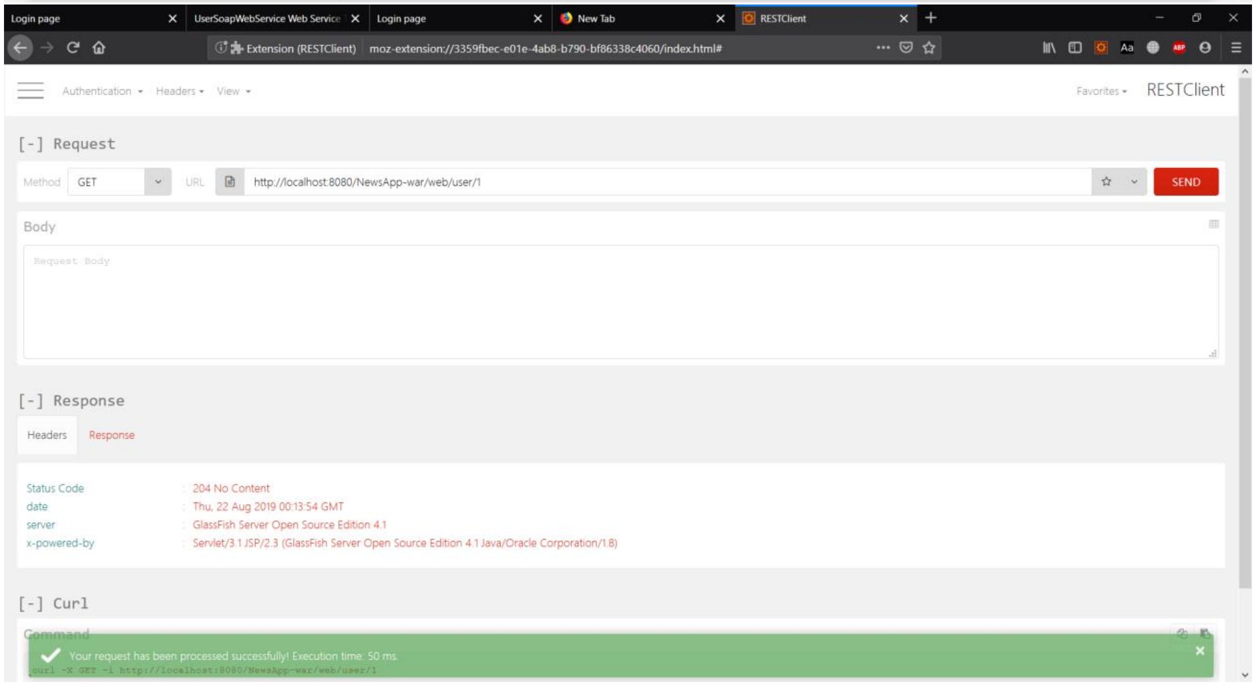
[-] Curl

Your request has been processed successfully! Execution time: 12 ms.


Method: DELETE
URL: http://localhost:8080/NewsApp-war/web/user/1

```
curl -X DELETE -i http://localhost:8080/NewsApp-war/web/user/1
```

Your request has been processed successfully! Execution time: 59 ms.

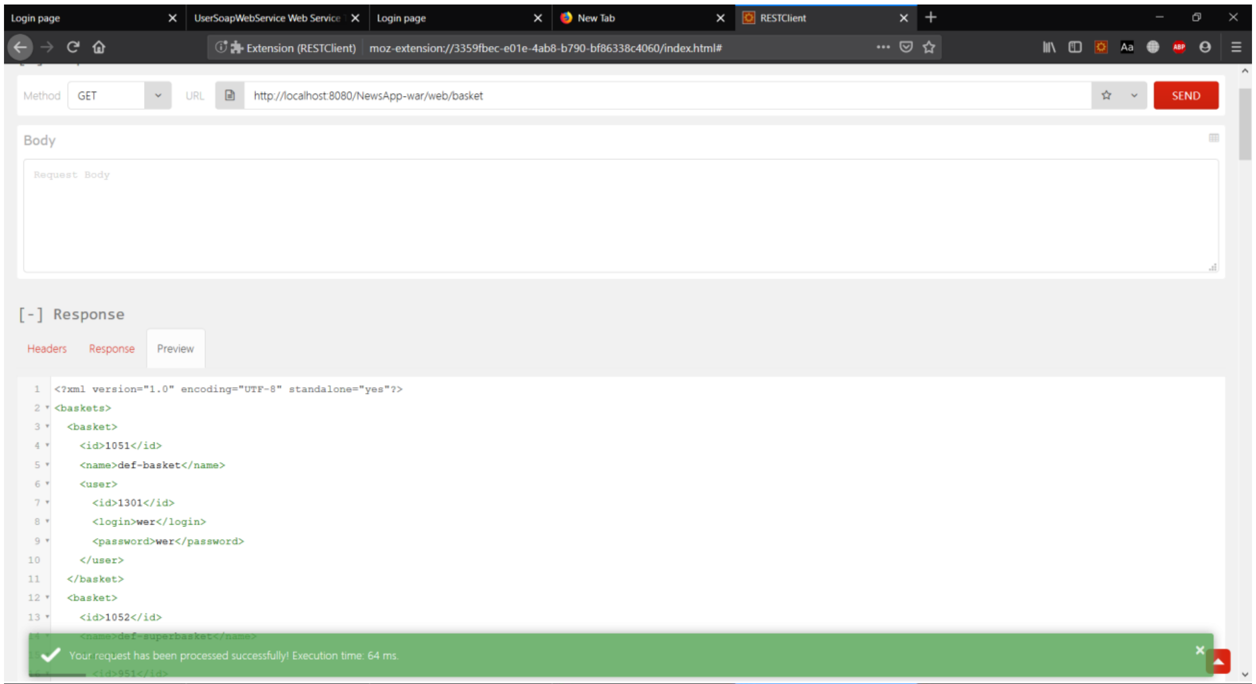

Method: GET
URL: http://localhost:8080/NewsApp-war/web/user/1

```
Status Code     : 204 No Content
date            : Thu, 22 Aug 2019 00:13:54 GMT
server          : GlassFish Server Open Source Edition 4.1
x-powered-by    : Servlet/3.1 JSP/2.3 (GlassFish Server Open Source Edition 4.1 Java/Oracle Corporation/1.8)
```

```
curl -X GET -i http://localhost:8080/NewsApp-war/web/user/1
```

Your request has been processed successfully! Execution time: 50 ms.

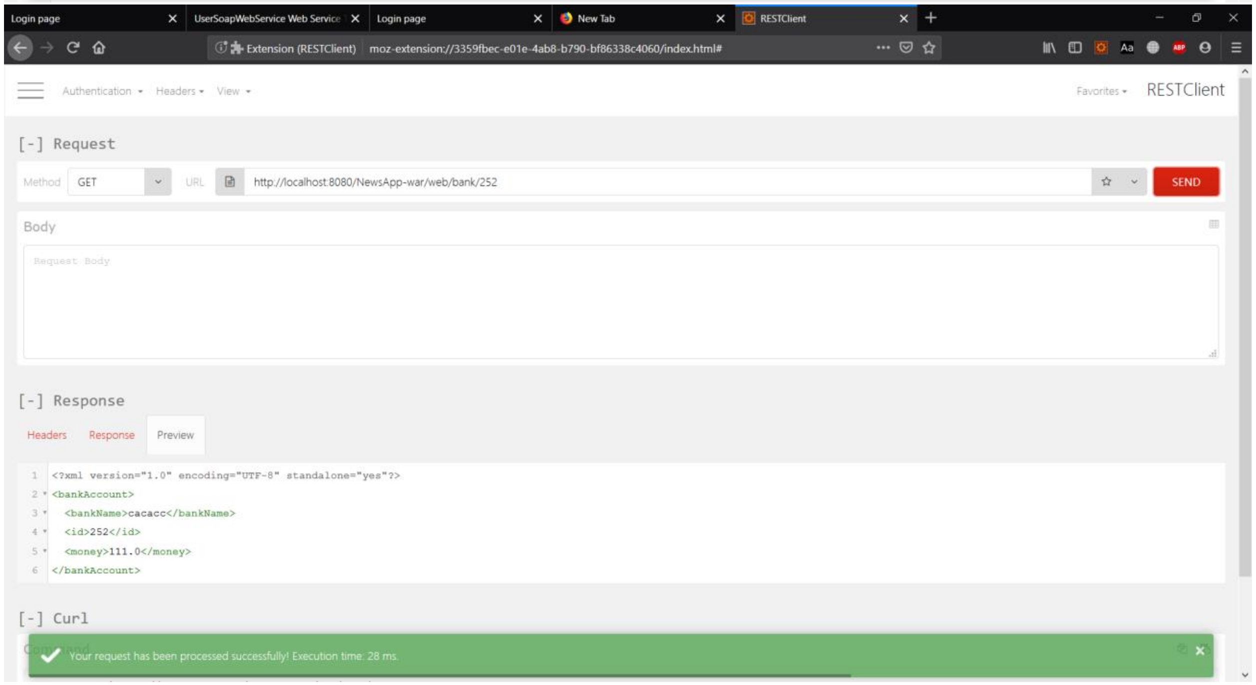
Process of showing and removing user.


Login page
UserSoapWebService Web Service
Login page
New Tab
RESTClient
Extension (RESTClient) moz-extension://3359fbec-e01e-4ab8-b790-bf86338c4060/index.html#
Body
[-] Response
Headers
Response
Preview
SEND

Method GET URL http://localhost:8080/NewsApp-war/web/basket

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<baskets>
  <basket>
    <id>1051</id>
    <name>def-basket</name>
    <user>
      <id>1301</id>
      <login>wer</login>
      <password>wer</password>
    </user>
  </basket>
  <basket>
    <id>1052</id>
    <name>def-superbasket</name>
      <id>951</id>
```

Your request has been processed successfully! Execution time: 64 ms.

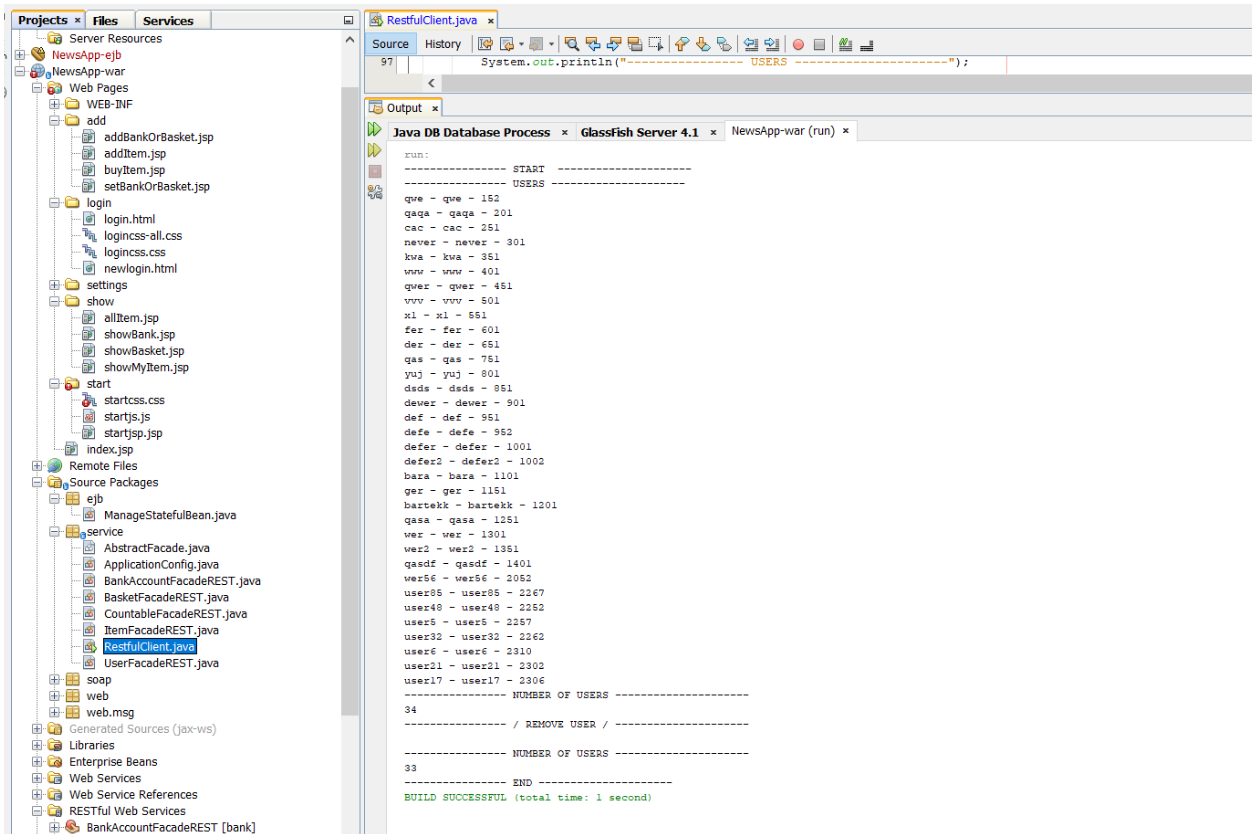


Method GET URL http://localhost:8080/NewsApp-war/web/bank/252

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<bankAccount>
  <bankName>cacacc</bankName>
  <id>252</id>
  <money>111.0</money>
</bankAccount>
```

Your request has been processed successfully! Execution time: 28 ms.

Process of retrieving information about particular basket.



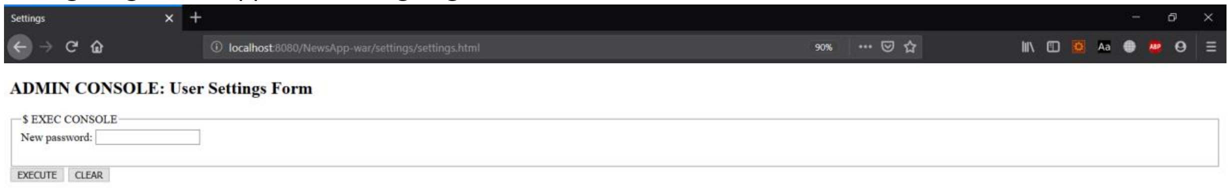
```
            System.out.println("---------------- USERS ---------------------");
```

```
run:
---------------- START  ---------------------
---------------- USERS ---------------------
qwe - qwe - 152
qaqa - qaqa - 201
cac - cac - 251
never - never - 301
kwa - kwa - 351
www - www - 401
qwer - qwer - 451
vvv - vvv - 501
xl - xl - 551
fer - fer - 601
der - der - 651
qas - qas - 751
yuj - yuj - 801
dsds - dsds - 851
dewer - dewer - 901
def - def - 951
defe - defe - 952
defer - defer - 1001
defer2 - defer2 - 1002
bara - bara - 1101
ger - ger - 1151
bartekk - bartekk - 1201
qasa - qasa - 1251
wer - wer - 1301
wer2 - wer2 - 1351
qasdf - qasdf - 1401
wer56 - wer56 - 2052
user85 - user85 - 2267
user48 - user48 - 2252
user5 - user5 - 2257
user32 - user32 - 2262
user6 - user6 - 2310
user21 - user21 - 2302
user17 - user17 - 2306
---------------- NUMBER OF USERS ---------------------
34
---------------- / REMOVE USER / ---------------------

---------------- NUMBER OF USERS ---------------------
33
---------------- END ---------------------
BUILD SUCCESSFUL (total time: 1 second)
```

Example of RestfulClient execution.

User can also change password (which is useful at the begging, since the default password is user's login) during usage of an app, without signing out.

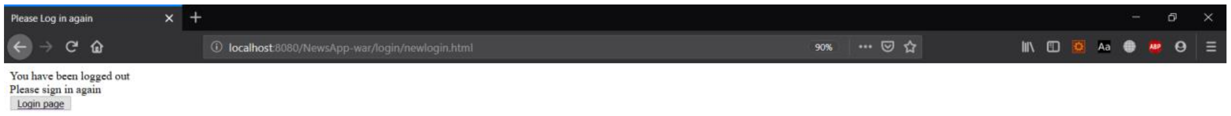


ADMIN CONSOLE: User Settings Form

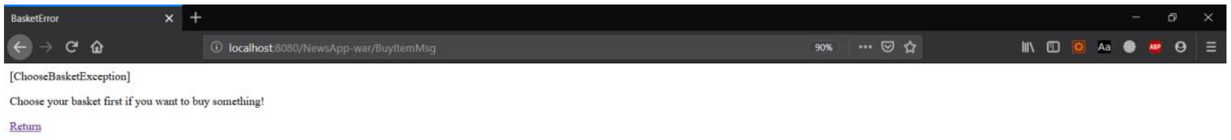
$ EXEC CONSOLE

New password:

EXECUTE CLEAR


You have been logged out
Please sign in again
Login page

By using the Log Out button (or Force Log out) – we can log out from an app in any moment. In that case, currentUser and currentBasket in EJB session maintenance bean would be cleared and nulled. If my try to reach any app page without log out – we are going to be redirected to page informed that we are not logged in. If me made some severe mistakes (such as buying item without selecting the basket) – we would be informed about particular exception which explains error and allows to return to page. There is no logout or server crash during that situation. All vulnerable action will be caught and do not disturbed server. Sometimes it may be necessary to refresh the page or log in again, but it also doesn't affect stored data.


[ChooseBasketException]

Choose your basket first if you want to buy something!

Return