Raport - Tema 5

PrivateSky structure

UserManager Asset:

- public attributes:
  - alias(string:alias): the alias of UserManager
  - users(map): a dictionary that will contain the users with key “userId” and value “user object”
- init:
  - params: alias(string)
  - set the alias and initialize “users” with an empty map
- addUser:
  - params: id(string), user(object)
  - adds the user with key “id” and value “user” in the map, if the key already exists in the map the value is updated
- removeUser:
  - params: id(string)
  - removes from the map the user with key “id”
- listUsers:
  - returns all the users from the dictionary
- getUserById:
  - params: id(string)
  - returns the user with key “id” if exists, otherwise returns Undefined

UserManagement Transaction:

- init:
  - params: alias(string)
  - searches in the blockchain if there exist an Usermanager with alias “alias”
  - initialize the UserManager
  - adds the user manager in the blockchain
- addUser:
  - params: alias(string), user(object)
  - searches in the blockchain for the Usermanager with alias “alias”
  - adds the user to the UserManager
  - commits the changes in the blockchain

- removeUser:
    - params: alias(string), id(string)
    - searches in the blockchain for the Usermanager with alias “alias”
    - removes the user with userId “id” to the UserManager
    - commits the changes in the blockchain
- listUsers:
    - params: alias(string)
    - searches in the blockchain for the Usermanager with alias “alias”
    - returns all the users stored in the Usermanager
- getUserById:
    - params: alias(string), id(string)
    - searches in the blockchain for the Usermanager with alias “alias”
    - returns the user with userId “id” from the UserManager

Client App

Initialisation:

```
const interact = pskclientRequire("interact");
interact.enableRemoteInteractions();
this.ris = interact.createRemoteInteractionSpace('testRemote',
     'http://127.0.0.1:8080', 'local/agent/example');
this.ris.startSwarm('UserManagement', 'init', 'usermanagement').onReturn(() =>
{
     console.log('SWARM started');
});
```

PrivateSky methods calls:

- `this.ris.startSwarm('UserManagement', 'listUsers', 'usermanagement')`
- `this.ris.startSwarm('UserManagement', 'getUserById', 'usermanagement', id)`
- `this.ris.startSwarm('UserManagement', 'addUser', 'usermanagement', user)`
- `this.ris.startSwarm('UserManagement', 'removeUser', 'usermanagement', id)`

Code:

```
$$.transaction.describe("UserManagement", {
   init: function (alias) {
       let transaction = $$.blockchain.beginTransaction({});
       let userManager = transaction.lookup('global.UserManager', alias);

       userManager.init(alias);

       try {
           transaction.add(userManager);
           $$.blockchain.commit(transaction);
       } catch (err) {
           this.return("UserManager creating failed!");
           return;
       }

       this.return(null, alias);
   },

   addUser: function (alias, user) {
       let transaction = $$.blockchain.beginTransaction({});
       let userManager = transaction.lookup('global.UserManager', alias);

       let result = userManager.addUser(user.id, user);

       try {
           transaction.add(userManager);
           $$.blockchain.commit(transaction);
       } catch (err) {
           this.return("Failed to save UserManager update!");
           return;
       }

       this.return(null, result);
   },

   removeUser: function (alias, userId) {
       let transaction = $$.blockchain.beginTransaction({});
       let userManager = transaction.lookup('global.UserManager', alias);

       let result = userManager.removeUser(userId);


       try {
           transaction.add(userManager);
```

```
            $$.blockchain.commit(transaction);
        } catch (err) {
            this.return("Failed to save UserManager update!");
            return;
        }

        this.return(null, result);
    },

    listUsers: function (alias) {
        let transaction = $$.blockchain.beginTransaction({});
        let userManager = transaction.lookup('global.UserManager', alias);

        let result = userManager.listUsers();

        this.return(null, result);
    },

    getUserById: function (alias, userId) {
        let transaction = $$.blockchain.beginTransaction({});
        let userManager = transaction.lookup('global.UserManager', alias);

        let result = userManager.getUserById(userId);

        this.return(null, result);
    }
});
```

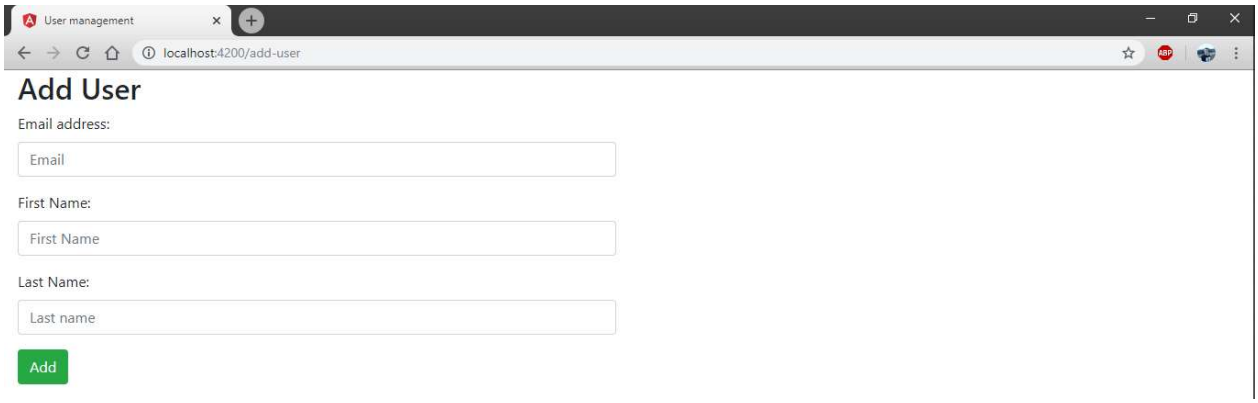
```
$$.asset.describe("UserManager", {
    public: {
        alias: "string:alias",
        users: "map",
    },

    init: function (alias) {
        this.alias = alias;

        if (!this.users) {
            this.users = {};
        }

        return true;
    },
```

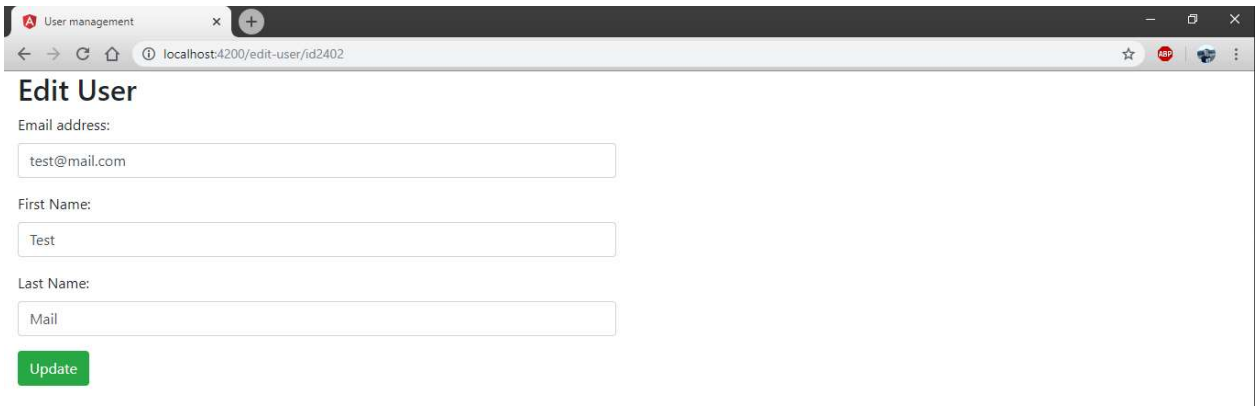
```
    addUser: function (id, user) {
        if (!this.users) {
            return false;
        }

        this.users[id] = user;

        return true;
    },

    removeUser: function (id) {
        if (!this.users) {
            return false;
        }

        this.users[id] = undefined;
        delete this.users[id];

        return true;
    },

    listUsers: function () {
        if (!this.users) {
            return false;
        }

        return Object.values(this.users);
    },

    getUserById: function (id) {
        if (!this.users) {
            return false;
        }

        return this.users[id];
    }
});
```

```
import { Injectable } from '@angular/core';
import { User } from "../model/user.model";
declare const pskclientRequire: any;

@Injectable()
export class UserService {
   ris: any;

   constructor() {
       const interact = pskclientRequire("interact");
       interact.enableRemoteInteractions();
       this.ris = interact.createRemoteInteractionSpace('testRemote',
'http://127.0.0.1:8080', 'local/agent/example');
       this.ris.startSwarm('UserManagement', 'init',
'usermanagement').onReturn(() => {
           console.log('SWARM started');
       });
   }

   getUsers(callback) {
       this.ris.startSwarm('UserManagement', 'listUsers',
'usermanagement').onReturn(callback);
   }

   getUserById(id: string, callback) {
       this.ris.startSwarm('UserManagement', 'getUserById', 'usermanagement',
id).onReturn(callback);
   }

   createUser(user: User, callback) {
       user.id = `id${Math.floor(Math.random() * 10000)}`;
       this.ris.startSwarm('UserManagement', 'addUser', 'usermanagement',
user).onReturn(callback);
   }

   updateUser(user: User, callback) {
       this.ris.startSwarm('UserManagement', 'addUser', 'usermanagement',
user).onReturn(callback);
   }

   deleteUser(id: string, callback){
       this.ris.startSwarm('UserManagement', 'removeUser', 'usermanagement',
id).onReturn(callback);
   }
}
```

App


User management
localhost:4200/list-users
User management
Add User
Id
FirstName
LastName
Email
Action
id2402
Test
Mail
test@mail.com
Edit
Delete
id2298
Andrei
Juravle
aijuravle@gmail.com
Edit
Delete


User management
localhost:4200/add-user
Add User
Email address:
Email
First Name:
First Name
Last Name:
Last name
Add


User management
localhost:4200/edit-user/id2402
Edit User
Email address:
test@mail.com
First Name:
Test
Last Name:
Mail
Update