|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A close up of a sign  Description automatically generatedIntern IDM  Java-task  Guide | |  | | |
|  |  |
|  |  |
| V.3 | |
| TESTED ON   * SYBASE ASE 16   Last Updated on   * 29.07.2022 | | |  | Technical Documentation Technical documentation for the IDM Onboarding project. |

Contents

[Changes 3](#_Toc110000886)

[1.Overview 4](#_Toc110000887)

* [1.1 Project Scope 4](#_Toc110000888)
* [1.2 Workflow 4](#_Toc110000889)

[2.Database design 5](#_Toc110000890)

* [2.1 Database Tables Overview 5](#_Toc110000891)

[3.Project structure 7](#_Toc110000892)

* [3.1 Class Overview 7](#_Toc110000893)

[4.Functionality Overview 8](#_Toc110000894)

* [4.1 Handler Methods Overview 8](#_Toc110000895)
* [4.2 Facade Methods Overview 17](#_Toc110000896)

# Changes



# 1.Overview

## 1.1 Project Scope

This IDM project is a REST based product which is used to simplify the organization of employee identities and managing roles and privileges. With the current release the following functionalities are available:

* Creation and management of employees
* Creation and management of roles and privileges
* Assigning, managing and removing access from the employees

The classes are structured around the Façade design pattern. SYBASE is used as a database and its accessed using data source.

## 1.2 Workflow

Below is an UML diagram representing the workflow of the program from start to end:

Diagram

Description automatically generated

# 2.Database design Diagram Description automatically generated

## 2.1 Database Tables Overview

Object\_Attributes

Used to store all types of attributes the objects can have.

* Id (int) – stores the id of the attribute
* Name (String) – stores the name of the attribute

Assigned-Attributes

Used to store all the attributes of a certain object.

* Id (int) – stores the id of the attribute
* object\_id (int) – stores the created object id
* attribute\_id (int) – stores the id of the type of attribute is assigned
* value (String) – stores the value of the attribute

Created-Objects

Used to store all the created objects in the system.

* Id (int) – stores the id of the created object
* object\_id (String) – stores the type of the created object
* name (String) – stores the uniqueId/uniqueName of the created object
* is\_deleted (String) – stores the sate the created object is

Relations

Used to store all the links between the objects, it is used for management of access of the certain object.

* this\_created\_object (int) - stores the id of the object we will assign access to
* other\_created\_object(int) – stores the id of the access we are assigning
* valid\_from (String) – stores the starting date of the access
* valid\_to (String) – stores the end date of the access
* parent (String) – stores the id of the original access from which this object is inherited

Objects

Used to store the type of objects we use.

* Id (int) – stores the id of the object
* name (String) – stores the name of the object

# 3.Project structureTimeline Description automatically generated with medium confidence

## 3.1 Class Overview

Object Info

Used to hold all the attributes an object can have, and it is used for e skeleton when you want to extract a certain object from the database with all its attributes.

Onboarding Handler

Used to store all the methods with the business logic used in the application. It makes works mostly with the façade methods by calling methods from them which allows the handler to work with data from the database.

User façade Local

Used to store queries which are intended for objects of type User. Holds all the methods that extract the information from the database and return it to the handler so it can work with that information.

Roles And Privileges Facad eLocal

Used to store queries which are intended for objects of type Role or Privilege. Holds all the methods that extract the information from the database and return it to the handler so it can work with that information.

Onboarding Recourse

Used to store all REST call’s which are used in the application. In each of the rest calls it calls a method with the respective logic needed from the handler method and that way it gets the needed information.

Abstract Resource

Used to store a ready for use Gson object

JsonE xception Mapper

Used to get error message and stack trace from REST call’s

# 4.Functionality Overview

In every of the method, we first check if the user, role or privilege is not deleted with a method that checks the objects attributes which indicates that.

## 4.1 Handler Methods Overview

Hire User

The method receives an object with all the attributes a user can enter himself and returns a String with the proper message. Validations are made for each of the fields and messages are returned. The user is created with a method from the user facade, and it is checked if he needs to be a manager or a regular user. After it’s called a method from the user facade again and each of the attributes is saved in the database.



**Here is an example of a create user method:**

** <- user is the object we originally put in the method**

**Here is an example of an addAttributes method:**

****

* “email” is the attribute we want to add
* User.getId() returns the id of the user
* “add” indicated that the attributed need to be added not updated. It also can be “update” when we want to update and already existing value of that user attribute. That functionality is used later in other methods.

**Here is an example of all the attributes the user can enter:**

**Text

Description automatically generated**

**Here is an example of all the existing attributes a user can have:**

**A picture containing text

Description automatically generated**

The fields uniqueId and email can’t be entered by the user, because they are autogenerated from the system when the user is created.

isDeleted indicates if the user is deleted or not. There are two options: “false”- the user is not deleted, “true”- the user is deleted.

Assignments is a list of all the roles or privileges (access) assigned to them.

Add Role/PRIVILEGE

The method receives an object with all the attributes a role or privilege can have and returns String with a proper message. Validations are made for each of the fields and messages are returned. The role or privilege is created with a method from the rolesAndPrivilegesFacade then all the attributes of the object are saved in the database.





**Here is an example of an addRole/Privilege method:**

** <- role is the object the method receives**

**<- privilege is the object the method receives**

**Here is an example of all existing attributes a role can have:**

**Text

Description automatically generated**

isDeleted indicates if the role was deleted or no. There are two options: “false”- the role is not deleted, “true”- the role is deleted

assignedTo is a list of all the users or roles assigned to it.

assignments are a list of all the roles or privileges that are assigned to the role

**Here is an example of all existing attributes a privilege can have:**

**Graphical user interface, text

Description automatically generatedGraphical user interface, text, application

Description automatically generated**

isDeleted indicates if the privilege was deleted or no. There are two options: “false”- the privilege is not deleted, “true”- the privilege is deleted

assignedTo is a list of all the users or roles that are assigned to the privilege

Modify User

The method receives an object with all the attributes a user can modify. Validations are made for each of the fields and proper messages are returned. The user is modified with a method from the facade.



**Here is an example of an addAttributes(update) method:**

****

“name” Is the attribute we want to modify

User.getId() returns the id of the user to the method

User.getName() returns the new attribute

“update” indicates that we want to modify an already existing value

**Here is an example of all attributes to a user that are modifiable:**

**Text

Description automatically generated**

Modify Role/Privilege

These methods receive object with all the attributes of the role/privilege that are modifiable. Validations are made and proper messages are returned of String type. The objects are modified with a method from userFacade.





**Here is an example of all the attributes of a role/privilege that are modifiable:**

**Graphical user interface, text, application, Word

Description automatically generated** **Graphical user interface, text, application, Word

Description automatically generated**

Leave

The method receives a uniqueId of type String and returns a String with a proper message.

The user is found with the uniqueId, and validations are made to them. If the user is a manager all the other users that are assigned to him are relocated to his manager. When deleted a user we remove all his ONLY privileges and mark him as a deleted one.



**Here is an example of a deleted user before and after:**

**Before After**

**Timeline

Description automatically generated** **Timeline

Description automatically generated with medium confidence**

Assign Access

The method receives a uniqueId of type String (the object we will assign access to), accessToAssign of type String (the access we are assigning) and from/to date of type String (the validity of the access). Validations are made then to the input.



**Here are all cases that are invalid in the assignAccess method:**

**Text

Description automatically generated**

In case we are assigning a role we check if the object we are assigning the role to is a user or another role. If it’s a user, we check if the user has the respective ONLY privileges for all the privileges the role might have in it and if the user has the ONLY privilege all the access of the role and role itself is assigned to the user. In case it’s a role, we skip the check for ONLY privilege and assign the role to the other role directly.

In case we are assigning a privilege we check if the object we are assigning the privilege to is a user or a role. If it’s a user, we check if the user has the respective ONLY privilege and if he has, we add the privilege to his access. In case it’s a role we check if we are assigning ONLY privilege and if we don’t we assign the privilege to the role.

Modify Access

The method receives a uniqueId of type String (the object we will modify the access), accessToAssign of type String (the access we are modifying) and from/to date of type String (the validity of the access). Validations are made then to the input.



**Here are all the cases that are invalid in the modifyAccess method:**

**Text, letter

Description automatically generated**

We can only modify access from validity perspective.

In case we want to modify the access with a role we get all the access of the role and modify their validity too.

In case we want to modify the access with a privilege we modify its validity only.

Display User/Role/Privilege

These methods receive a uniqueId/uniqueName of the object we want to get, finds it in the database with a method from the facade and return it.







Display All Users/roles/Privileges

These methods do not receive any input, they only return a list of all the respective objects by getting them from the database with a method from the facade and return it.





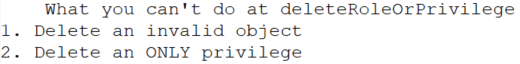


Delete Role Or Privilege

The method receives a name of type String and returns a proper message of type String.



**Here are all the cases that are invalid in the deleteRoleOrPrivilege method:**

****

In case it’s a privilege, we check if it’s an ONLY privilege, if it’s not we mark it as a deleted one.

In case it’s a role, we mark the role as a deleted one

Remove Access

The method receives a uniqueId of type String (the object we will remove the access from) and an access of type String (the access we will remove)



**Here are all the cases that are invalid in the removeAccess method:**

**Text, letter

Description automatically generated**

First, we check if we are removing a role or a privilege. In case it’s a role we get all the access from the role that Is assigned to the respective object and remove it aswell. In case it’s a privilege we check if it’s an ONLY privilege and if it is we remove all the other privileges from the same system as the ONLY privilege aswell, if it’s not we only remove the privilege.

Email Generator

This method receives a username of type String and returns an email of type String for the user following this format:



Text

Description automatically generated

IS USER LEFT

This method receives an object and checks his isDeleted attribute and returns a boolean as a respond.



CHECK FOR ONLY PRIVILEGE

This method receives one list of objects onlyPrivilegesForObject (containing all ONLY privileges of an object) and another list of objects privilegesForObject (containing all the privileges for an object) and returns a boolean as a respond. The first cycle gets the current ONLY privilege from the first list. The second cycle gets the current privilege from the second list. A check is made if the ONLY privilege has the same system as the privilege. If it’s true, the privilege is removed from the second list.

At the end it’s checked if the size of the second list is not zero. That way if all the privileges from the second list had the same system as the ONLY privileges from the first list the access would be valid and the method will return “true”, otherwise there would have been left some privileges from different system and the access would be invalid and the method will return “false”.



Display All Access For User

This method receives a name of type String and returns a list of objects containing all the access (even the deleted one) of that user.



Display All Managers

This method returns a list of all users that are marked as managers in the system.



## 4.2 Facade Methods Overview

Find all Manager Users

The method returns a list of all users in the system that are marked as managers. They are found with specifying the number of the attribute id that stays for managerId and also specifying that it should be true, and the user should not be deleted.



Find Manager By Id

The method receives a managerId of type String and returns the user with that id who needs to be a manager.



Find All Users By Manager Id

The method receives a managerId of type String and returns a list of all the users that have his manager id in their managerId attribute.



Create User

The method receives a user and stores its unique Id and object type in the nv\_created\_objetcs table and sets the id of the created object



Add Role

The method receives a role and stores its object type and unique name in the nv\_created\_objects table, then stores the created role id in the object



Add Privilege

The method receives a privilege and stores its object type and unique name in the nv\_created\_objects table, then stores the created privilege id in the object



Add Attributes

The method receives an attribute of type String (the attribute we will assign), id of type int (the id of the object we will assign the attribute), an action of type String (the value of the attribute we will assign) and action (add or update).

If the action is “add” it is used the Insert query, otherwise it’s used the update query.

Depending on the attribute id it is chosen with what id the value should be inserted/updated.



Find Object By Unique Id

The method receives a uniqueId of type String and looks for matches in the nv\_created\_objects table. The result can be any type of object. When it finds a match, it returns the object with the respective uniqueId/uniqueName.



Find Object By Id

The method receives an id of type int and looks for a match in the nv\_assigned\_attributes table. The result can be of any type of object. After it finds a match, it returns all the attributes of the object with that id.



Find All Users By Object Id

The method receives an id of type int and looks for all relations with users for that object id. It returns a list of users of type ObjectInfo with all their attributes.



Check Access

The method receives an id of type int and another id2 of type int and looks for matches for those ids in the nv\_relations table. It returns a boolean response. If it’s true it indicates that there is such access, if it returns false it indicates that there isn’t such access



Find All Access For Object

The method receives an id of type int and looks for all the relations with that object in the nv\_relations\_table. It returns a list with all the related objects.



Find All Available Access For Object

The method receives an id of type int and looks for all the relations with that object in the nv\_relations\_table. It returns a list with all the related objects which are not deleted.



Update Validity Access

The method receives an id of type int (the id of the object we will update the access), another id2 of type int (the id of the access we will be updating), and the value of the dates of type String. The values are updated in the query and results are stored in the nv\_relations table.



Assign Access

The method receives an id of type int (the id of the object we will assign access), another id2 of type int (the id of the access we will be assigning), and the value of the dates of type String. The values are assigned in the query and results are stored in the nv\_relations table.



Find All Users

The method returns a list of type ObjectInfo which stores all objects of type user in the system.



Attach Attributes To Object

The method receives a rSet of type ResultSet predefined in previous methods and an object (the object we will be attaching the attributes). The attributes are attached to the object and at the end the object is returned with all his attributes.



Unique Identifier Generator

The method receives a user and generates a uniqueId for him following this pattern:



Graphical user interface, text, application

Description automatically generated

Id Setter

The method receives a user and attaches his id from the database to the object



Set Is Manager

The method receives a user and sets his isManager attribute to “true”



Get DB CONNECTION

The method makes connection with the database using DataSource

