Par défaut la gestion des transactions est gérée par le container EJB qui s'occupe des appels à commit et rollback et les méthodes définies dans le DAO servent/sont/fixent les points de démarcations.

Nos DAO sont des stateless session bean (SLSB), ceci est particulièrement important du point de vue de la démarcation des transactions. Le conteneur lance une transaction chaque fois qu'une méthode est appelée et commit si la méthode renvoie un résultat. Si la méthode lève une exception, le conteneur annule/rollback la transaction.

Dans notre implémentation, le seul moment où une requête intervient sur plusieurs DAO en même temps avec des appels en "cascade" est le cas de suppression d'un user.

Nous avons décidé que dans ce cas, les applications liées à cet utilisateur sont réassignées à un autre utilisateur disabled qui récupère et stock ces applications dans le cas où le développeur souhaite revenir sur sa décision

On pourrait aussi implémenter une feature avec par exemple un choix lors de la suppression de l'utilisateur si l'on veut garder les applications ou non.

Toujours en est-il que du coup niveau transactionnel, si la réassignation des projets échouent, on préfère rollback le tout avant la suppression de l'utilisateur.

Le comportement par défaut correspond à celui recherché comme prouvé avec l'expérience suivante :

On voit bien que la méthode deleteUser de UserDAO appelle une méthode reassignProjetOfUser d'un autre DAO ProjectDAO. Par défaut le transaction sera "unique/globale" pour la méthode deleteUser et si une exception est levée dans la méthode reassign, la totalité de la méthode sera rollback et donc le user existe toujours et ses projets lui sont toujours assignés.

```java
//@TransactionAttribute(TransactionAttributeType.REQUIRES_NEW)
public boolean deleteUser (String email) {
    boolean ok = false;
    try {
        Connection conn = database.getConnection();
        PreparedStatement ps = conn.prepareStatement( sql: "DELETE FROM " + TABLE_NAME + " WHERE email = ?;");
        ps.setString( parameterIndex: 1, email);

        // Check SQL Execution
        if (ps.executeUpdate() == 0) {
            ps.close();
            conn.close();
            throw new SQLException("Delete Failed");
        }

        projectDAO.reassignProjectOfUser(email);

        ps.close();
        conn.close();
        ok = true;
    } catch (SQLException e) {
        e.printStackTrace();
        ok = false;
    }
    return ok;
}
```

```java
@Override
public boolean reassignProjectOfUser(String email){
    boolean ok = true;

    try {
        PreparedStatement ps = database.getConnection().prepareStatement(
                sql: "UPDATE " + TABLE_NAME_JOIN +
                        " SET email = '" + EMAIL_ASSIGN +
                        "' WHERE email = ?;");
        ps.setString( parameterIndex: 1, email);


        throw new RuntimeException("Reassign failed - rollback test");

        //ps.close();

    } catch (SQLException e) {
        e.printStackTrace();
        ok = false;
    }
    System.out.println("[ProjectDAO - reassignProjectOfUser] return - " + ok);
    return ok;
}
```

Modification	firstname	lastname	email	password	address	zip	country	admin	enable	reset
modifier	backup	backup	backup@backup.backup	backup	backup	backup	backup	0	0	0
modifier	Dylan	Hamel	compte.dylan.hamel@gmail.com	123123	Rue de Savoie 3B	1196	Suisse	0	1	0
modifier	testFN	testLN	test@test.test	testPWD	testAdd	testZIP	testCountry	0	1	0

modifier	26	compte.dylan.hamel@gmail.com	dylan1

Dans le cas où l'on change le type de transaction à requires_new le résultat sera différent, en effet une transaction indépendante sera faite pour chaque appel à la BD donc la suppression de l'user sera commit avant l'appel au reassignProject qui lui va lever une exception et être rollback, du coup l'utilisateur n'existera plus mais les projets lui seront toujours assignés malgré tout.

```java
@TransactionAttribute(TransactionAttributeType.REQUIRES_NEW)
public boolean deleteUser (String email) {
    boolean ok = false;
    try {
        Connection conn = database.getConnection();
        PreparedStatement ps = conn.prepareStatement( sql: "DELETE FROM " + TABLE_NAME + " WHERE email = ?;");
        ps.setString( parameterIndex: 1, email);

        // Check SQL Execution
        if (ps.executeUpdate() == 0) {
            ps.close();
            conn.close();
            throw new SQLException("Delete Failed");
        }

        projectDAO.reassignProjectOfUser(email);

        ps.close();
        conn.close();
        ok = true;
    } catch (SQLException e) {
        e.printStackTrace();
        ok = false;
    }
    return ok;
}
```

☐ modifier	26	compte.dylan.hamel@gmail.com	dylan1

☐ Modification	firstname	lastname	email	password	address	zip	country	admin	enable	reset
☐ modifier	backup	backup	backup@backup.backup	backup	backup	backup	backup	0	0	0
☐ modifier	testFN	testLN	test@test.test	testPWD	testAdd	testZIP	testCountry	0	1	0

Dans le cas où aucune exception n'est levée, ses projets sont bien ré-assignés au user backup@backup.backup et l'utilisateur est supprimé.

https://examples.javacodegeeks.com/enterprise-java/ejb3/transactions/ejb-transaction-management-example/