

ORM in Java & Postgres

SWE3



September 24, 2021

FH Technikum – BIF5

Chahed Rajoub (if19b166)

Table of Contents

[Used languages: 2](#_Toc93344032)

[Functionality: 2](#_Toc93344033)

[Data Model: 2](#_Toc93344034)

[**Manual:** 3](#_Toc93344035)

ORM Functionality:

## Used languages:

* Java
* Postgres

## Functionality:

* A Good ORM can lead to much less code and much less repetition in a project
* Allows the changes in Database technologies
* Queries should be generated by the mapper without even thinking about Dialect or how the transaction is happening.
* This Design is simplified to approach a simple application of OR Mapper using Java, Postgres Container in Docker
* Database tables are already present, and the Mapper enables the queries on these tables
* The mapper is designed Code first style following these steps:
  + Configuring the connection with the Database as settings in persistence layer
  + Adding the classes as tables and the properties as columns in Entity layer
  + Adding different functions 'Select, delete, insert, update, drop' in Services layer
  + Adding transactions services 'join, find' in Service layer
  + Model layer is used to show a demo of the ORMapper and uses annotations
  + Annotations layer to define the reflection function that will define the tables and the columns

## Data Model:

This ORM supports tables that has foreign keys to refer to other tables and connect to them. as an example for 1-n relationship:

Table

Description automatically generated with medium confidence

as an example, for n-m relationship:

‘ Diagram

Description automatically generated

**Manual:**

The following steps explain how to use this ORM:

1. Create a Postgres Database Docker Container. It could be #something like this:

*$ docker run --name Base -p 5432:5432 -e POSTGRES\_PASSWORD=if19b166 -d postgres*

1. Adding Url, username, password to Config.xml in Settings folder.
2. Create a new ORM object where it is needed.
3. Create classes as entities with annotations to connect to the tables in DB

@table -----------> for a class

@Column ----------> for a property

@PrimaryKey-------> for unique ids

@ForeignKey-------> for foreign ids

@IgnoreProperty---> to ignore a property

1. ORM can make the calls to our database and save them in lists of objects.
2. The following functions can be used to fulfill the needs of this ORM:
   1. SelectAllRows
   2. SelectAllColumns
   3. SelectByID
   4. SelectbyColumn
   5. Insert
   6. JoinByForiegnKey
   7. JoinAll
   8. DeleteRowbyId
   9. Update