Advanced Databases & noSQL (INFDEV03-5)
Assignment 1

Instructions

- The assignment must be submitted within Friday of Week 5.
- The assignment must be implemented using Java with Eclipselink ORM.
- Not delivering the assignment in time implies taking the exam at the retake.

Assignment

This assignment consists in defining a database to record user data for a *Massive Multiplayer Online Role Playing Game* (MMORPG) and to build a small application for the user to subscribe and log in. When a user wants to register to the server he is asked to pick a unique user name. The system allows the user to recharge money from his bank account which are stored in the database as `Balance`. The user can use his money to pay the subscription. The minimum subscription is one month for 5€, two months for 8€, three months for 10€, and 35€ for a whole year. The system stores the name, surname, IBAN, the date of the last payment, the months bought with the last payment, the character slots, the login password, and whether the use was banned from the game. Money can be used also to unlock additional slots for characters (see below). A player receives 5 character slots with the subscription, and can buy additional slots for 1€ each.

Each player can create multiple characters on different servers up to their character slots amount. Each character has a unique name which is displayed in-game, an owner which is the name of the user who created it, the server address where the character was created, a character class, a race, and the level of the character in the game.

A server is identified by a unique address, a name, a location which is the area of the server, the max users that can create be online, and the users currently online.

Given this requirements, the relational schema of the database is the following:

characters			
name	class	race	level

users				
user_name	balance	first_name	last_name	iban
character_slots	last_payment	months_payed	password	banned

servers				
address	name	location	max_users	connected_users

owns	
name	user_name

stores	
address	user_name

- In the relation `owns` the attribute `user_name` is a foreign key to `user_name` in the relation `user`, and `name` a foreign key to `name` in the relation `characters`.
- In the relation `stores` the attribute `address` is a foreign key to `address` in the relation `servers`, and `user_name` is a foreign key to `user_name` in the relation `users`.
- The underlined attributes in each relation form the primary key for that relation.

The student is asked to implement the database schema into PostgreSQL and to develop a user application in Java using hibernate ORM. The user application is divided into three modules: (*i*) the user login screen, (*ii*) the user management screen, and (*iii*) the characters management screen. The application should contain a graphical interface for the login phase:

- Register a new account: the registration is allowed only if the chosen user name has not been picked by anyone else otherwise the program should ask to input a different username. The user should fill in all the fields required to create a record for the user table.
- Login to an existing account: if the username does not exist the program should notify the user. The same happens if the user inputs a wrong password for an existing username.

Once the login/registration phase has succeeded you are brought to a window divided into two sections: the user management section, and the character management section. It should be possible to switch between sections from the user interface.

The user management section should allow to perform the following operations:

- Add money to the user account. This is done immediately without having to transfer money from an actual bank account (do not try to connect to ING servers, please!).

- Renovate the subscription. The user should be able to choose to pay for 1, 2, 3 months, or one year. Update the last payment and the subscription data accordingly. Of course this should be possible only if the user has enough money loaded into his account, otherwise the application should notify that he needs to load more money into his account.

- Buy more character slots. The user chooses how many extra slots he wants to buy.

The character management section should allow the user to perform the following operations:

- Create a new character. This should be possible only if the user has a enough free slots (i.e. he does not own more characters than his available slots) and if the provided character name is unique. The user should be able to pick a character class and a race (you are free to invent whatever classes and races you want). Just for testing initialize the character level with a random number between 1 and 100 (usually it should start from level 1). The program should also remove a free slot for the user.

- Select a character from a dropdown menu. When the character is selected the interface should show all the data about that character. The characters must be ordered from the highest level to the lowest and only the characters owned by the current user should be visible. Do the sorting in the database and not in the Java program.

- Connect to a server. This can be done only if the server is not full. The program notifies if the connection is successful (the server has enough free slots), or if it fails (the server is full). When a player connects the connected players field must be updated accordingly.

Note: the server table should be initialized from Postgres with server records. Insert some servers before testing the user application (you are free to invent the data for them).