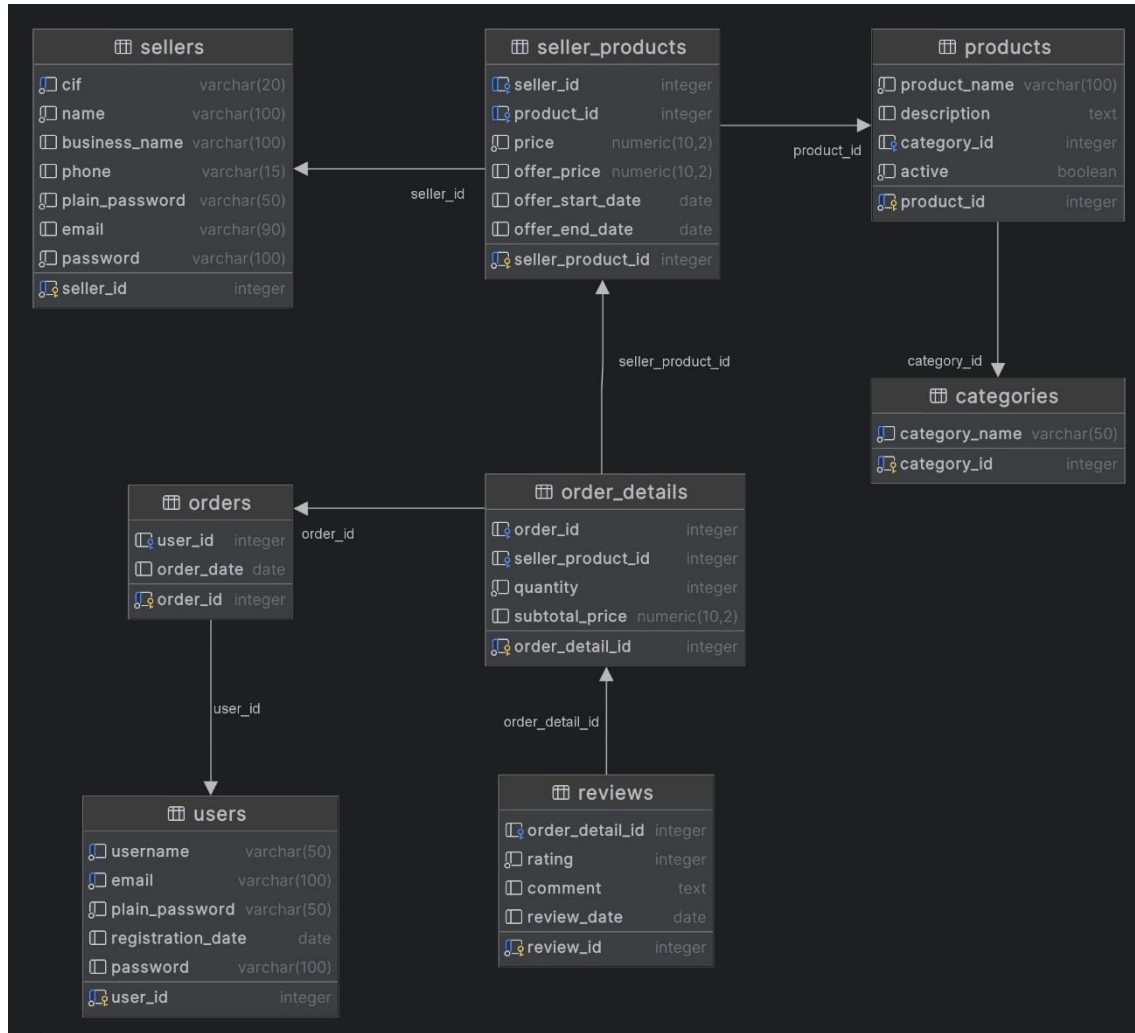


Final Activity
First Period

Online Market
Sellers

Final activity: online market sellers

Using the provided script (online-market-2425.sql), the student must generate the database that will fit the following e-r schema:



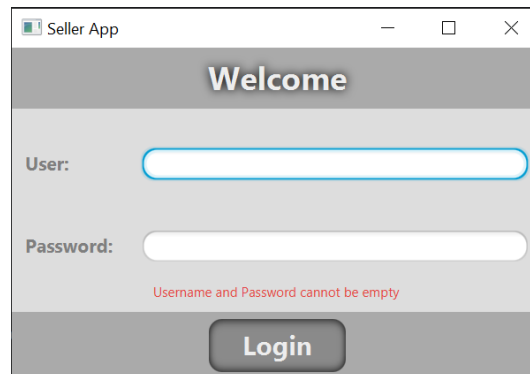
For the purposes of this activity, we will only use four tables: Sellers, seller_products, products and categories. We have to develop an application that should permit a seller registered in our online market to sell a product from the list of previously available products.

1. User Login

The first thing to do will be to log the seller into the app. As you can see in the image above, the 'Sellers' table have two 'password' fields, the 'plain_password' field (the password stored in a readable format) and the 'password' field (is the MD5 of the previous field).

Our application should ask for a combination of user (seller CIF) and password and check if the pair is present in the database. If it is, then the main view will be presented to the

seller. If not, it will be informed that an error has occurred. Here you can see an example of the login screen:



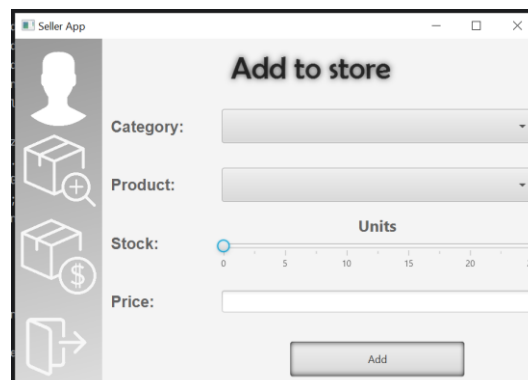
Once logged, the seller will be allowed to do the following actions:

2. Modify his/her personal data



Keep in mind that, if the field has a formatted value, it should be checked. Also, every mandatory field must be introduced. The CIF field shouldn't be modified. When any field does not comply with the requirements, the user must be informed. Finally, you should ensure that the password field has been correctly typed.

3. Add a product from the list of products to his/her store.



Sellers can add products to their own personal store from the list of products. The stock (quantity) of the product to add must be specified. A product can't be added twice. To

ensure this, you must write **a stored procedure/function** that returns the list of products not present yet in the store list.

4. Make an offer on a product of his/her store.

The screenshot shows a web form titled "Add offer". On the left is a vertical sidebar with four icons: a person, a box with a plus sign, a box with a dollar sign, and a box with an arrow. The main form area contains the following fields: "Product:" with a dropdown menu, "From:" with a text input and a small calendar icon, "To:" with a text input and a small calendar icon, and "Discount:" with a text input. At the bottom right of the form is a button labeled "Add".

Only one product can have a discounted price in a time period. The discounted price must match the following requirements:

- For a 30-day period: maximum discount of 10%
- For a 15-day period: maximum discount of 15%
- For a 7-day period: maximum discount of 20%
- For a 3-day period: maximum discount of 30%
- For a 1-day period: maximum discount of 50%

HIBERNATE TIPS

- The database uses generators to automatically add the key field in almost every table. If you experience any problem inserting (null values in the key field due to non-automatically generated values) you may have to change the `@GeneratedValue` annotation to `@GeneratedValue(generator="name-of-the-generator")`. You can check generator names in the script. For example, the generator for the id of the table 'sellers' is 'sellers_seller_id_seq'.
- Fields with null values can cause you problems, even in the case that you have a default value in the table. A bypass for this is use the `@DynamicInserts` annotation, that will skip null values when doing an insert. You can also use `@DynamicUpdate` (to skip updating non-changed fields). Another option is to create a constructor for the Entity/POJO with the same default values you have in the table.
- When mapping the relationships, keep in mind whether you are using the field for updating or just reading, in order to map the field as foreign key or as a class. Remember that you can also map the both of them, but annotating the property as `insertable=false, updatable=false`.
- Entities (POJOs) can be manually upgraded with your own functionality. Remember, in this case, to annotate every method that is not going to be persisted -stored- with the `@Transient` annotation.