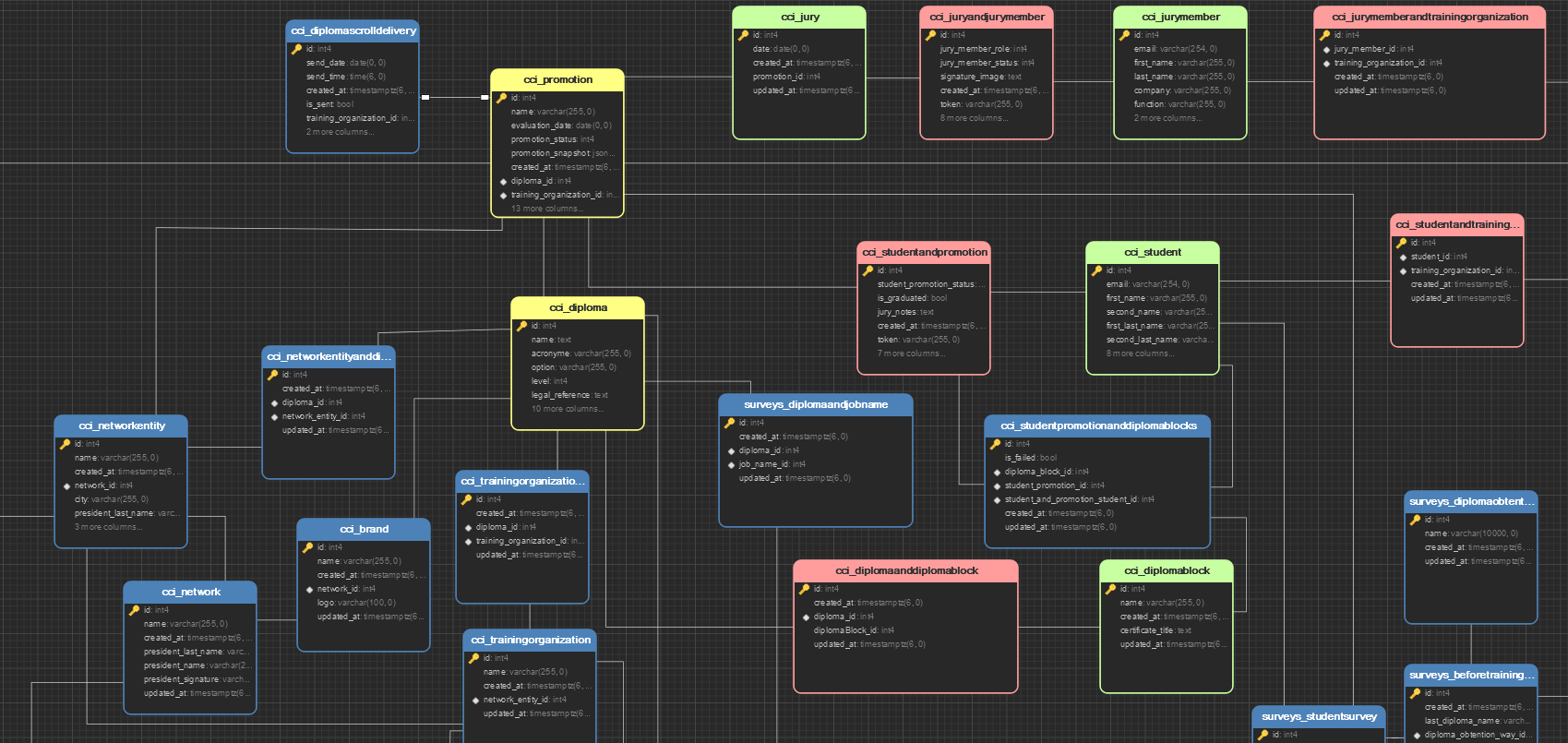
**Documentation of the refactoring of the many-to-many to many-to-one relationship.**

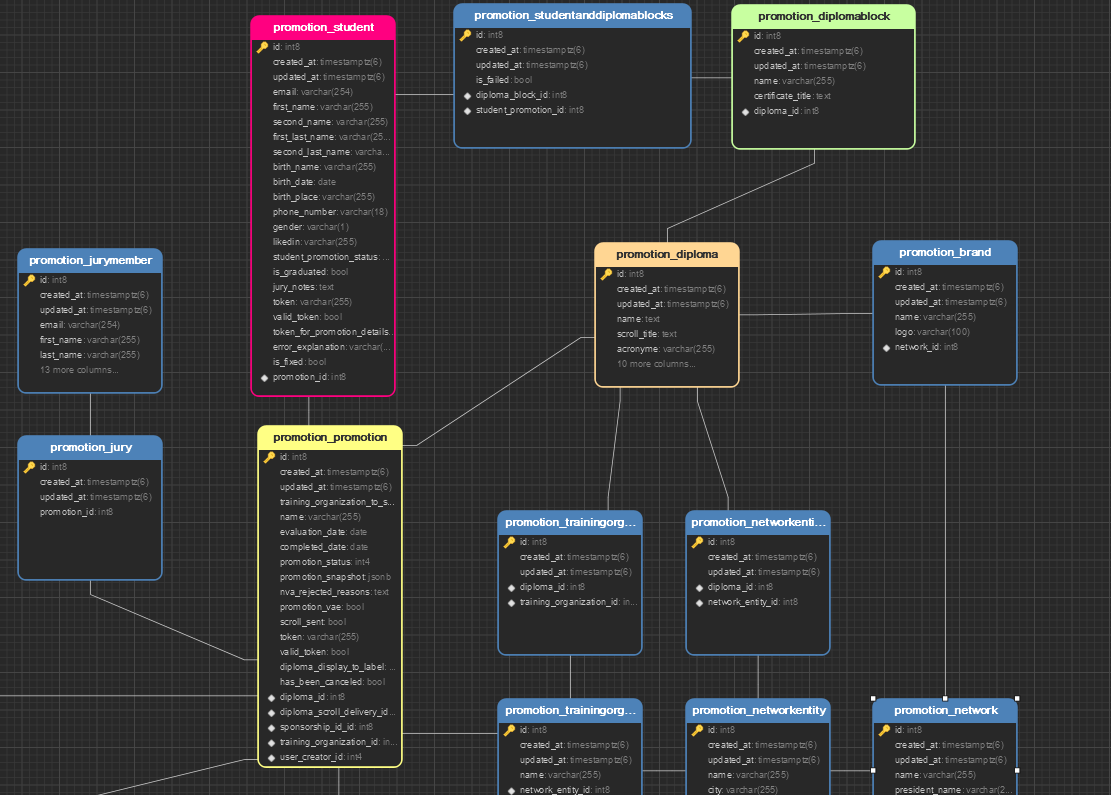
**Introduction**

We describe the impact and the time it will take to refactor the Diplomation Project, to change the many-to-many relationship to many-to-one in some tables. Eliminating some redundancies and gaining agility in the execution of the project.

Currently, the database is made up of the following tables::



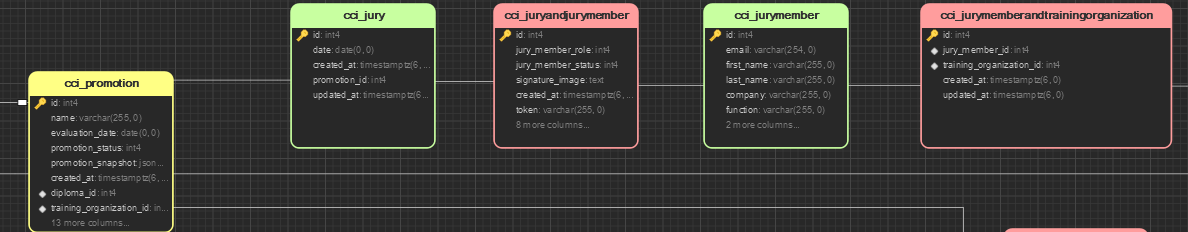
The tables to be deleted are shown in red in the image. Resulting in a resulting database as shown in the following image:



**Changes to be made**

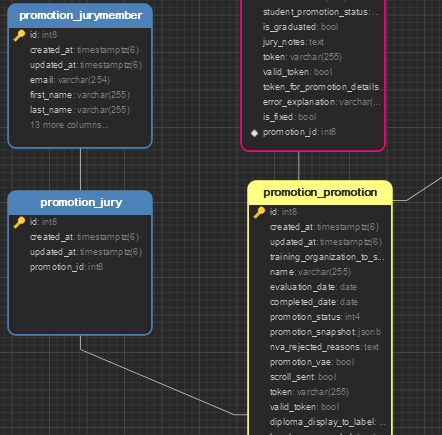
First of all, you must make a backup copy of your data to make sure you do not lose any information in the process. These changes will be separated into several phases. Among the planned ones are the Promotion relationship with Jury and then the Promotion relationship with Student.

**1. Promotion with Jury relationship:**

This part of the business has a structure as shown in the image. To make a change in it, changes must be made in the model.

**Changes in the model and in the database.**

1. Create a temporary table with the same attributes of the tables that are colored red and JuryMember.
2. Populate the temporary table with the data found in the table colored red and the JuryMember table.
3. Delete the red and JuryMember tables.
4. Rename the temporary table to JuryMember. To continue with the same naming pattern.

All this change generates a relationship in database with the following form:

**Code changes**

After the above changes will have suffered in the names of model classes and instances that will no longer exist. Then, as step two it would be necessary to change all the names and classes that made use of the tables that were eliminated. And restructure the views that make use of these models.

**Time estimation**

It is estimated that the total time to perform this refactoring:

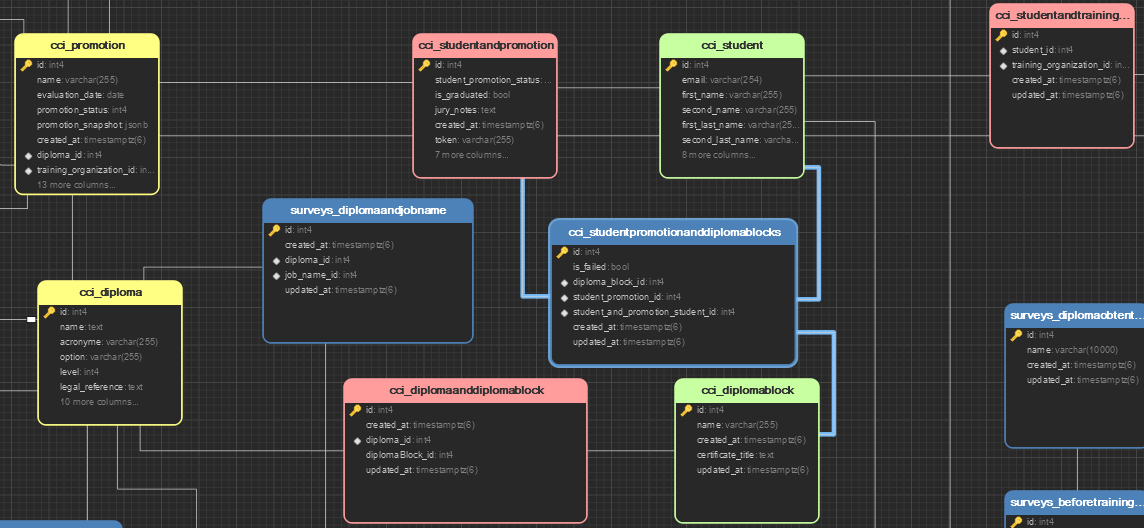
For a favorable case:

* 16 hours, to change the database structure.
* 20 hours, to make the code changes and restructure the endpoints.

For an unfavorable case:

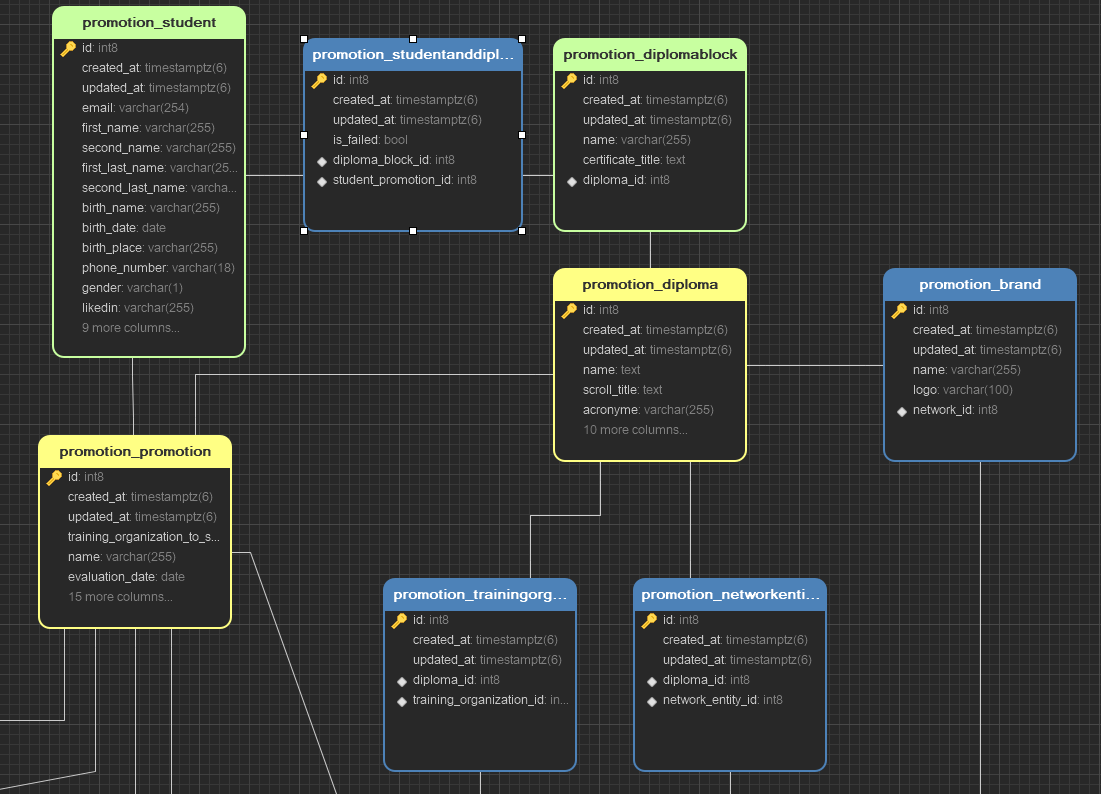
* 24 hours, to fix bugs .
* 24 hours, to fix errors in the views and files related to the business.

**2. Promotion relationship with Studiants:**

Initially, in this phase the database structure with Promotion and Students relationship is as follows:

**Changes in the model and in the database.**

1. Create a temporary table with the same attributes of the Student and StudentAndPromotion tables.
2. Populate the temporary table with the data found in the Student and StudentAndAndPromotion tables.
3. Delete tables mentioned above.
4. Eliminate the relation NtoN of Student with TrainingOrganization. Since this relation exists with Promotion and the students already belong to a promotion.
5. Change the name of the temporary table to Student. This way you will have in a single table all the data related to a student.
6. Create another temporary table with all the attributes of DiplomaBlock and with a Many to One relationship with Diploma.
7. Populate the temporary table with the DiplomaBlock data.
8. Delete the DiplomaBlock table.
9. Change the name of the temporary table to DiplomaBlock.

These changes will appear as follows in the database:

**Changes in the code**

As in the previous phase, there are model class names that will no longer exist and attributes or data that will not be found in another model class. In the same way, it would be necessary to change the names of classes, instances and restructure the views referring to these modified model classes.

**Time estimation**

Total estimated time to perform this refactoring:

For a favorable case:

* 18 hours, to change the database structure.
* 23 hours, to make the code changes and restructure the endpoints.

For an unfavorable case:

* 24 hours, to fix bugs .
* 24 hours, to fix errors in the views and files related to the business.