# 1. Rules

- Students will take the test individually.
- You can use the code developed in class (and anything that fits in a pendrive)
- The code created for this test will be uploaded to moodle as a single file obtained executing the command `git archive --format zip --output ../testX.zip master` (where X is character that identifies the Test)
- The zip file will include a text file (name it `report.txt`) describing the step realized in this test.

## Enunciado

A hospital wants to implement a drug management service. This service will be supported by the database corresponding to the following relational schema

patient(id , nameP)
doctor(id,nameD)
prescription(id,doctor(id)⇑,patient(id)⇑,)
where ⇑ means foreign key.

Remember that Django adds by default an attribute named *id* that works as primary key.

## Tasks to do

1. Create a project named `project` in Django. Then create an application named `application`. Data will be persisted in a postgres database called `exam`.

2. Create the data model (`models.py`) and configure the project so that the data can be seen using the admin interface (`http://localhost:8000/admin/`).

3. Write a script (called `populate.py`) that insert the following data into the data-set using the Django API.

   doctor(1,'doctor1')
   doctor(2,'doctor2')
   doctor(3,'doctor3')
   doctor(4,'doctor4')

   patient(1,'patient1')
   patient(2,'patient2')

   prescription(1,1,1)
   prescription(2,2,1)
   prescription(3,1,2)
   prescription(4,2,2)
   prescription(5,3,2)

4. Deploy the project in Heroku and populate the data base using the script `populate.py`, doublecheck that the admin interface is available and can be accessed with the username/password `alunodb`. Nota: write down in the file `memoria.txt` the admin interface URL. IMPORTANT: Whenever the project was been deployed in heroku (and always before the exam finalization) show it to your teacher.

5. Using the template `prescription.html`, crea a new web page with URL $PROJECT_URL/application/prescription/ that returns a list with the last 3 prescriptions. Note: Do not modify the file `prescription.html`. If the database is empty the web page should show an error message using the variable `error`. The last 3 prescriptions should be obtained using a query, that is, do not get a list with all prescriptions and then sort it in the view.

6. Create a test (which should be run with the command `python manage.py aplication.test --keepdb`) that:

   - deletes all doctors, patients and prescriptions
   - creates the doctor (1,'doctor1')
   - creates el patient (1,'patient1')
   - creates la prescription (1,1,1)
   - creates la prescription (2,1,1)
   - creates la prescription (3,1,1)
   - creates la prescription (4,1,1)

- access to the view $PROJECT_URL/application/prescription/
- using assertions, check that the returned prescriptions are correct.

## 2. Grading Criteria

5 points if the following criteria are met: after executing the commands:

```
  dropdb -U alumnodb -h localhost examen
createdb -U alumnodb -h localhost examen
python manage.py makemigrations application
python manage.py migrate
python manage.py createsuperuser
python ./poblar.py
```

It is possible to access to the admin interface at URL `http://localhost:8000/admin/` and the data shown is the one described in the previous section, subsection (3). The database used must be postgres.

7 points if the following criteria are met:

- The criteria listed in the previous paragraph are fully satisfied
- The application has been deployed in Heroku, it is possible to access to the admin interface using the username/password `alumnodb` and the data shown is the one described in the previous section, subsection (3).
- the code deployed in Heroku must be IDENTICAL to the one uploaded to moodle

8 points if the following criteria are met:

- The criteria listed in the previous paragraph are fully satisfied
- The page $PROJECT_URL/application/prescription/ works properly

10 points if the following criteria are met:

- The criteria listed in the previous paragraph are fully satisfied
- The page test satisfies all the requirements

## A. Templates

```
<html>
 <head>
 </head>
<body>
```

```
{% if error %}
   {{ error }}
{% endif %}
<table>
{% for prescription in prescriptions %}
        <tr>
            <th>{{ prescription.id }}</th>
            <td>{{ prescription.patient.nombreP }}</td>
            <td>{{ prescription.doctor.nombreM }}</td>
        </tr>
{% endfor %}
</table>
</body>
</html>
```