



Entrenamiento de Python-Django - 2

ORM

Resumen: Hoy abordaremos el ORM de Django.

Versión: 1

Contenido

I	Reglas generales	2
II	Reglas específicas de hoy	3
III	Ejercicio 00	5
IV	Ejercicio 01	6
V	Ejercicio 02	7
VI	Ejercicio 03	9
VII	Ejercicio 04	11
VIII	Ejercicio 05	13
Ejercicio IX	06	15
X	Ejercicio 07	17
XI	Ejercicio 08	19
XII	Ejercicio 09	21
XIII	Ejercicio 10	23
XIV	Presentación y evaluación por pares	25

Capítulo I

Reglas generales

- Su proyecto debe realizarse en una máquina virtual.
- Su máquina virtual debe tener todo el software necesario para completar su proyecto. Estos software deben configurarse e instalarse.
- Puede elegir el sistema operativo que utilizará para su máquina virtual.
- Debe poder utilizar su máquina virtual desde una computadora del clúster.
- Debe utilizar una carpeta compartida entre su máquina virtual y su máquina host.
- Durante tus evaluaciones utilizarás esta carpeta para compartir con tu repositorio.

Sus funciones no deben cerrarse inesperadamente (fallo de segmentación, error de bus, doble liberación, etc.), salvo por comportamientos indefinidos. Si esto ocurre, su proyecto se considerará no funcional y recibirá un 0 en la evaluación.

Te animamos a crear programas de prueba para tu proyecto, aunque este trabajo no tenga que entregarse ni se califique. Esto te permitirá evaluar fácilmente tu trabajo y el de tus compañeros. Estas pruebas te resultarán especialmente útiles durante la defensa. De hecho, durante la defensa, puedes usar tus propias pruebas o las del compañero que estás evaluando.

Envía tu trabajo al repositorio Git asignado. Solo se calificará el trabajo en el repositorio Git. Si Deepthought se encarga de calificar tu trabajo, lo hará después de la evaluación por pares. Si se produce un error en alguna sección de tu trabajo durante la calificación de Deepthought, la evaluación se detendrá.

Capítulo II

Reglas específicas de hoy

- Debe utilizar un intérprete de Python3.

Cada ejercicio será independiente. Si alguna de las características requeridas ya se ha abordado en ejercicios anteriores, duplíquela en el ejercicio actual.

- Debes trabajar en una base de datos postgresql llamada djangotraining y crear un rol llamado djangouser, cuya contraseña será "secret", que tendrá todos los derechos sobre el mismo.

- La carpeta de tu repositorio debe ser un proyecto de Django. El proyecto debe tener el nombre de... el día actual.

- Utilizaremos el concepto de aplicación de Django para separar los ejercicios:
Los ejercicios de hoy deben estar ubicados en una aplicación Django específica que lleve el nombre del ejercicio correspondiente y se encuentre en la raíz del repositorio.

- El proyecto Django debe estar configurado correctamente para cumplir con los requisitos del ejercicio.
No podrá cambiar las configuraciones durante la evaluación.

- No podrás entregar ninguna migración con tu trabajo.

- En cada ejercicio en el que se mencione ORM, debes usar el ORM de Django.
No debe escribir ninguna línea de SQL.

- En cada ejercicio en el que se mencione SQL, debes utilizar la biblioteca psycopg2 y ejecutar todas las solicitudes en SQL.

A continuación se muestra un ejemplo de estructura típica para el repositorio de un estudiante llamado krichard, sobre el Día d42, incluye 2 ejercicios:


```
|-- krichard
| |-- .
| |-- ..
| |-- .git
| |-- .gitignore
| |-- d42
| |-- __init__.py
| |-- configuraciones.py
| |-- urls.py
| |-- wsgi.py
| |-- ex00
| |-- admin.py
| |-- apps.py
| |-- formularios.py
| |-- __init__.py
| |-- modelos.py
| |-- pruebas.py
| |-- urls.py
| |-- vistas.py
| |-- ex01
| |-- admin.py
| |-- apps.py
| |-- formularios.py
| |-- __init__.py
| |-- modelos.py
| |-- pruebas.py
| |-- urls.py
| |-- vistas.py
| |-- administrar.py
```



Sé inteligente: factoriza tu código y hazlo fácil de usar. Ahorrarás.
a veces.

Capítulo III

Ejercicio 00

	Ejercicio 00
Ejercicio 00: SQL - construcción de una tabla	
Directorio de entrega: ex00/	
Archivos a entregar:	
Funciones permitidas:	

Cree una aplicación Django llamada ex00 así como una vista interna a la que se debe acceder en la siguiente URL: 127.0.0.1:8000/ex00/init.


Esta vista debe crear una tabla SQL en PostgreSQL con la ayuda de la biblioteca psycopg2 y devolver una página con el mensaje "OK" si la operación es correcta. De lo contrario, debe devolver un mensaje de error que describa el problema.

La tabla SQL debe ajustarse a esta descripción:

- Debe llamarse ex00_movies.
- Debe crearse sólo si aún no existe.
- Sólo debe incluir los siguientes campos:
 - título: cadena de caracteres única y variable, tamaño máximo de 64 bytes, no nulo.
 - episode_nb: completo, CLAVE PRINCIPAL.
 - opening_crawl: texto, puede ser nulo, sin límite de tamaño.
 - director: cadena de caracteres variable, no nula, tamaño máximo 32 bytes.
 - productor: cadena de caracteres variable, no nula, tamaño máximo 128 bytes.
 - release_date: fecha (sin hora), no nula.

Capítulo IV

Ejercicio 01

	Ejercicio 01
Ejercicio 01: ORM - construcción de una tabla	
Directorio de entrega: ex01/	
Archivos a entregar:	
Funciones permitidas:	


Cree una aplicación llamada ex01. En ella, cree un modelo de Django llamado Movies con estos campos exactamente:

- título: cadena de caracteres única y variable, tamaño máximo de 64 bytes, no nulo.
- episode_nb: completo, CLAVE PRINCIPAL.
- opening_crawl: texto, puede ser nulo, sin límite de tamaño.
- director: cadena de caracteres variable, no nula, tamaño máximo 32 bytes.
- productor: cadena de caracteres variable, no nula, tamaño máximo de 128 bytes.
- release_date: fecha (sin hora), no nula.

Este modelo también debe redefinir el método `__str__` para que devuelva el título. atributo.

Capítulo V

Ejercicio 02

	Ejercicio 02
Ejercicio 02: SQL - Inserción de datos	
Directorio de entrega: ex02/	
Archivos a entregar:	
Funciones permitidas:	

Debe crear una aplicación Django llamada ex02. En esta aplicación, las vistas son accesibles mediante las siguientes URL:

- 127.0.0.1:8000/ex02/init: debe crear una tabla con las mismas especificaciones que la requerida para ex00 excepto que se llamará ex02_movies.

Si la operación es correcta, debe devolver una página con el mensaje "OK". De lo contrario, debe mostrar un mensaje que describa el problema.

- 127.0.0.1:8000/ex02/populate: debe insertar los siguientes datos en la tabla creada por la vista anterior:

- Número de episodio: 1 - Título: La amenaza fantasma - Director: George Lucas - Productor: Rick McCallum - Fecha de lanzamiento: 19/05/1999
- episodio_nb: 2 - título: El ataque de los clones - director: George Lucas - productor: Rick McCallum - fecha de lanzamiento: 16/05/2002
- número de episodio: 3 - título: La venganza de los Sith - director: George Lucas - productor: Rick McCallum - fecha de lanzamiento: 19/05/2005
- número de episodio: 4 - título: Una nueva esperanza - director: George Lucas - productor: Gary Kurtz, Rick McCallum - fecha de lanzamiento: 25/05/1977
- número de episodio: 5 - título: El Imperio Contraataca - director: Irvin Kershner - productores: Gary Kurtz, Rick McCallum - fecha de lanzamiento: 17/05/1980

- Número de episodio: 6 - Título: El Regreso del Jedi - Director: Richard Marquand - Productores: Howard G. Kazanjian, George Lucas, Rick McCallum - Fecha de estreno: 25/05/1983
- Número de episodio: 7 - Título: El despertar de la fuerza - Director: JJ Abrams - Productores: Kathleen Kennedy, JJ Abrams, Bryan Burk - Fecha de lanzamiento: 11/12/2015


Debe devolver una página con el mensaje "OK" tras cada inserción exitosa. De lo contrario, debe mostrar un mensaje de error que indique el problema.

- 127.0.0.1:8000/ex02/display: debe mostrar todos los datos incluidos en la tabla ex02_movies en una tabla HTML, incluidos los eventuales campos vacíos.

Si no hay datos disponibles o hay un error, la página simplemente debe mostrar "No hay datos". disponible".

Capítulo VI

Ejercicio 03

	Ejercicio 03
Ejercicio 03: ORM - inserción de datos	
Directorio de entrega: ex03/	
Archivos a entregar:	
Funciones permitidas:	

Crea una nueva aplicación Django llamada ex03. Dentro de ella, crea un modelo Django idéntico a el creado en el ex01.

Esta aplicación debe incluir las vistas accesibles a través de las siguientes URL:

- 127.0.0.1:8000/ex03/populate: debe insertar el modelo de esta aplicación, el siguientes datos:
 - Número de episodio: 1 - Título: La amenaza fantasma - Director: George Lucas - Productor: Rick McCallum - Fecha de lanzamiento: 19/05/1999
 - episodio_nb: 2 - título: El ataque de los clones - director: George Lucas - productor: Rick McCallum - fecha de lanzamiento: 16/05/2002
 - número de episodio: 3 - título: La venganza de los Sith - director: George Lucas - productor: Rick McCallum - fecha de lanzamiento: 19/05/2005
 - número de episodio: 4 - título: Una nueva esperanza - director: George Lucas - productor: Gary Kurtz, Rick McCallum - fecha de lanzamiento: 25/05/1977
 - número de episodio: 5 - título: El Imperio Contraataca - director: Irvin Kershner - productores: Gary Kurtz, Rick McCallum - fecha de lanzamiento: 17/05/1980
 - Número de episodio: 6 - Título: El Regreso del Jedi - Director: Richard Marquand - Productores: Howard G. Kazanjian, George Lucas, Rick McCallum - Fecha de estreno: 25/05/1983

- Número de episodio: 7 - Título: El despertar de la fuerza - Director: JJ Abrams - Productores: Kathleen Kennedy, JJ Abrams, Bryan Burk - Fecha de lanzamiento: 11/12/2015

Debe devolver una página con el mensaje "OK" tras cada inserción exitosa. De lo contrario, debe mostrar un mensaje de error que indique el problema.

- 127.0.0.1:8000/ex03/display: debe mostrar todos los datos incluidos en las películas tabla en una tabla HTML, incluidos los eventuales campos vacíos.


Si no hay datos disponibles o hay un error, la página debe mostrar simplemente "No hay datos disponibles".



Durante la evaluación, se realizará la migración antes de la pruebas.

Capítulo VII

Ejercicio 04

	Ejercicio 04
Ejercicio 04: SQL - Eliminación de datos	
Directorio de entregas: ex04/	
Archivos a entregar:	
Funciones permitidas:	

Cree una aplicación llamada ex04. Debe contener las vistas accesibles mediante las siguientes URL:

- 127.0.0.1:8000/ex04/init: debe crear una tabla con las mismas especificaciones que la requerida para la aplicación ex00 excepto que se llamará ex04_movies.

Debe devolver una página con el mensaje "OK" tras cada inserción exitosa. De lo contrario, debe mostrar un mensaje de error que indique el problema.

- 127.0.0.1:8000/ex04/populate: debe insertar los datos descritos en el ejercicio ex02 en la tabla creada en la vista anterior.

Esta vista debe volver a insertar cualquier dato eliminado.

Debe devolver una página con el mensaje "OK" tras cada inserción exitosa. De lo contrario, debe mostrar un mensaje de error que indique el problema.

- 127.0.0.1:8000/ex04/display: debe mostrar todos los datos incluidos en la tabla ex04_movies en una tabla HTML, incluidos los eventuales campos vacíos.

Si no hay datos disponibles o hay un error, la página simplemente debe mostrar "No hay datos". disponible".

- 127.0.0.1:8000/ex04/remove: debe mostrar un formulario HTML que contenga una lista desplegable de títulos de películas y un botón de envío llamado eliminar.


Los títulos de las películas son los contenidos en la tabla `ex04_movies`.

Cuando se valida el formulario, la película seleccionada se elimina de la base de datos y el formulario se vuelve a mostrar con la lista actualizada que contiene las películas restantes.

Si no hay datos disponibles o hay un error, la página simplemente debe mostrar "No hay datos". disponible".

Capítulo VIII

Ejercicio 05

	Ejercicio 05
Ejercicio 05: ORM - Eliminación de datos	
Directorio de entregas: ex05/	
Archivos a entregar:	
Funciones permitidas:	

Crea una nueva aplicación Django llamada ex05. Crea dentro de ella un modelo idéntico al de ex01.

Esta aplicación debe incluir las vistas accesibles a través de las siguientes URL:

- 127.0.0.1:8000/ex05/populate: debe insertar los datos descritos en el ejercicio ex03 en la aplicación creada.

Esta vista debe volver a insertar cualquier dato eliminado.

Debe devolver una página con el mensaje "OK" tras cada inserción exitosa. De lo contrario, debe mostrar un mensaje de error que indique el problema.

- 127.0.0.1:8000/ex05/display: debe mostrar todos los datos incluidos en las películas tabla en una tabla HTML, incluidos los eventuales campos vacíos.

Si no hay datos disponibles o hay un error, la página debe mostrar simplemente "No hay datos disponibles".

- 127.0.0.1:8000/ex05/remove: debe mostrar un formulario HTML que contenga una lista desplegable de títulos de películas y un botón de envío llamado eliminar.

Los títulos de las películas son los que figuran en el modelo Películas de esta aplicación.

Cuando se valida el formulario, la película seleccionada se elimina de la base de datos y el formulario se vuelve a mostrar con la lista actualizada que contiene las películas restantes.


Si no hay datos disponibles o hay un error, la página simplemente debe mostrar "No hay datos". disponible".



Durante la evaluación, se realizará la migración antes de la pruebas.

Capítulo IX

Ejercicio 06

	Ejercicio 06
Ejercicio 06: SQL - Actualización de datos	
Directorio de entregas: ex06/	
Archivos a entregar:	
Funciones permitidas:	

Cree una nueva aplicación de Django llamada ex06. Debe contener las vistas accesibles mediante las siguientes URL:

- 127.0.0.1:8000/ex06/init: debe crear una tabla con las mismas especificaciones que la requerida para la aplicación ex00 excepto que se llamará ex06_movies e incluirá los siguientes campos adicionales:
 - creó un tipo de fecha y hora (fecha y hora) que, cuando se crea, debe establecerse automáticamente en la fecha y hora actuales.
 - actualizó un tipo datetime (fecha y hora), que, cuando se crea, debe establecerse automáticamente en la fecha y hora actuales y se actualiza automáticamente con cada actualización gracias al siguiente disparador:

```

CREAR O REEMPLAZAR FUNCIÓN update_changestamp_column()
DEVUELVE EL DISPARADOR COMO
$$
COMIENZA NUEVO.actualizado =
ahora(); NUEVO.creado = VIEJO.creado;
DEVUELVE NUEVO;
FIN; $
$ idioma 'plpgsql'; CREAR DISPARADOR
update_films_changestamp ANTES DE ACTUALIZAR EN ex06_movies PARA CADA FILA EJECUTAR
PROCEDIMIENTO update_changestamp_column();

```

- 127.0.0.1:8000/ex06/populate: debe rellenar la tabla creada en la vista anterior con los datos descritos en el ejercicio doit peupler la table ex02.

Debe devolver una página con el mensaje "OK" tras cada inserción exitosa. De lo contrario, debe mostrar un mensaje de error que indique el problema.

- 127.0.0.1:8000/ex06/display: debe mostrar todos los datos incluidos en ex06_movies tabla en una tabla HTML.

Si no hay datos disponibles o hay un error, la página simplemente debe mostrar "No hay datos disponibles".


- 127.0.0.1:8000/ex06/update: debe gestionar el envío y recepción de un formulario.

Este último debe permitir seleccionar una película en un menú desplegable que contiene las películas incluidas en la tabla ex06_movies y escribir texto en el segundo campo. Al validar el formulario, la vista debe reemplazar el campo opening_crawl de la película seleccionada con el texto escrito en el formulario de la tabla ex06_movies.

Si no hay datos disponibles o hay un error, la página debe mostrar simplemente "No hay datos disponibles".

Capítulo X

Ejercicio 07

	Ejercicio 07
Ejercicio 07: ORM - Actualización de datos	
Directorio de entregas: ex07/	
Archivos a entregar:	
Funciones permitidas:	

Crea una nueva aplicación Django llamada ex07. Crea un modelo idéntico al de ex01 dentro de ella, excepto que agregarás los siguientes campos:

- creó un tipo de fecha y hora (fecha y hora) que, al crearse, debe establecer en la fecha y hora actuales.
- se actualizó un tipo de fecha y hora (fecha y hora) que, cuando se crea, debe establecerse automáticamente en la fecha y hora actuales y se actualiza automáticamente con cada actualización.

Esta aplicación debe contener las vistas accesibles en las siguientes URL:

- 127.0.0.1:8000/ex07/populate: rellena el modelo previamente creado con los mismos datos que ex02.

Debe devolver una página con el mensaje "OK" tras cada inserción exitosa. De lo contrario, debe mostrar un mensaje de error que indique el problema.

- 127.0.0.1:8000/ex07/display: muestra todos los datos incluidos en la tabla Películas en una tabla HTML.

Si no hay datos disponibles o hay un error, la página debe mostrar simplemente "No hay datos disponibles".

- 127.0.0.1:8000/ex07/update: debe gestionar el envío y recepción de un formulario. Este último debe permitir elegir una película en un menú desplegable que contiene las películas.

incluido en la tabla Películas y escribir texto en el segundo campo.

Al validar el formulario, la vista debe modificar el campo `opening_crawl` de la película seleccionada con el texto escrito en el formulario del modelo Películas.


Si no hay datos disponibles o hay un error, la página debe mostrar simplemente "No hay datos disponibles".



Durante la evaluación, se realizará la migración antes de la pruebas.

Capítulo XI

Ejercicio 08

	Ejercicio 08
Ejercicio 08: SQL - Clave externa	
Directorio de entregas: ex08/	
Archivos a entregar:	
Funciones permitidas:	

Cree una nueva aplicación de Django llamada ex08. Esta aplicación debe contener las vistas accesibles mediante las siguientes URL:

- 127.0.0.1:8000/ex08/init: debe crear dos tablas.

El primero debe llamarse ex08_planets e incluir los siguientes campos:

- id: serial, clave principal
- nombre: cadena de caracteres única y variable, tamaño máximo de 64 bytes, no nulo.
- clima: cadena de caracteres variables.
- diámetro: entero.
- periodo orbital: entero.
- población: gran conjunto.
- periodo_de_rotación: entero.
- agua superficial: real.
- terreno: cadena de caracteres variable, tamaño máximo 128 bytes.

El segundo debe llamarse ex08_people e incluir los siguientes campos:

- id: serial, clave principal.
- nombre: cadena de caracteres única y variable, tamaño máximo de 64 bytes, no nulo.

- birth_year: cadena de caracteres variable, tamaño máximo de 32 bytes.
 - género: cadena de caracteres variable, tamaño máximo de 32 bytes.
 - eye_color: cadena de caracteres variable, tamaño máximo de 32 bytes.
 - hair_color: cadena de caracteres variable, tamaño máximo de 32 bytes.
 - altura: entera.
 - masa: real.
 - homeworld: cadena de caracteres variable, tamaño máximo de 64 bytes, clave externa, que hace referencia a la columna de nombre de la tabla 08_planets.
- 127.0.0.1:8000/ex08/populate: debe rellenar ambas tablas copiando el contenido de los archivos people.csv y planets.csv en las tablas correspondientes, respectivamente: ex08_personas y ex08_planetas.

Esta vista debe devolver una página con el mensaje "OK" cada vez que se inserte correctamente. De lo contrario, debe mostrar un mensaje de error que indique el problema.
 - 127.0.0.1:8000/ex08/display: muestra todos los nombres de los personajes, su mundo natal así como el clima, que es ventoso o moderadamente ventoso, ordenados en orden alfabético del nombre del personaje.


Si no hay datos disponibles o hay un error, la página simplemente debe mostrar "No hay datos". disponible".



Obtenga información sobre el método `psycopg2copy_from`

Capítulo XII

Ejercicio 09

	Ejercicio 09
Ejercicio 09: ORM - Clave externa	
Directorio de entregas: ex09/	
Archivos a entregar:	
Funciones permitidas:	

Crea una nueva aplicación Django llamada ex09 y crea dos modelos dentro. El primero... se llamará Planetas y contendrá los siguientes campos:

- nombre: cadena de caracteres única y variable, tamaño máximo de 64 bytes, no nulo.
- Clima: cadena de caracteres variables.
- diámetro: entero.
- periodo orbital: entero.
- población: gran conjunto.
- periodo_de_rotación: entero.
- agua superficial: real.
- Terreno: cadenas de personajes.
- creó un tipo de fecha y hora (fecha y hora) que, al crearse, debe establecer en la fecha y hora actuales.
- se actualizó un tipo de fecha y hora (fecha y hora) que, cuando se crea, debe establecerse automáticamente en la fecha y hora actuales y se actualiza automáticamente con cada actualización.

Este modelo también debe redefinir el método `__str__()` para que devuelva el nombre atributo.

El segundo modelo que crearás debe llamarse Personas y contener los siguientes campos:

- nombre: cadena de caracteres, tamaño máximo de 64 bytes, no nulo.
- birth_year: cadena de caracteres, tamaño máximo 32 bytes.
- género: cadena de caracteres, tamaño máximo 32 bytes.
- eye_color: cadena de caracteres, tamaño máximo de 32 bytes.
- hair_color: cadena de caracteres, tamaño máximo de 32 bytes.
- altura: entera.
- masa: real.
- homeworld: cadena de caracteres, tamaño máximo de 64 bytes, clave externa que hace referencia a la columna de nombre de la tabla Planetas de esta aplicación.
- creó un tipo de fecha y hora (fecha y hora) que, al crearse, debe establecer en la fecha y hora actuales.
- se actualizó un tipo de fecha y hora (fecha y hora) que, cuando se crea, debe establecerse automáticamente en la fecha y hora actuales y se actualiza automáticamente con cada actualización.

Este modelo también debe redefinir el método `__str__()` para que devuelva el nombre atributo.

En esta aplicación, también crearás una vista que debe ser accesible en el siguiente anuncio:
vestido: 127.0.0.1:8000/ex09/display.

Esta vista debe mostrar todos los nombres de los personajes, su mundo natal, así como el clima, que es ventoso o moderadamente ventoso, ordenados en orden alfabético del nombre del personaje en una tabla HTML.

Si no hay datos disponibles, la vista debe mostrar el siguiente texto: "No hay datos disponibles, utilice la siguiente línea de comando antes de usar:" seguido de una línea de comando.

Esta línea de comando debe ser la que se ejecute desde la raíz de tu repositorio para poder insertar todos los datos incluidos en el archivo `ex09_initial_data.json` (provisto con los recursos de hoy) en los modelos previamente creados.


Deberás proporcionar estos archivos con tu repositorio.



Durante la evaluación, se realizará la migración antes de la pruebas.

Capítulo XIII

Ejercicio 10

	Ejercicio 10
Ejercicio 10: ORM - Muchos a Muchos	
Directorio de entregas: ex10/	
Archivos a entregar:	
Funciones permitidas:	

Crea una nueva aplicación Django llamada ex10 y crea 3 modelos dentro:

- Planetas y Personas: Ambos modelos deben ser idénticos a los del ex09.
- Películas: Este modelo debe ser idéntico al del ex01 excepto que debes agregar los caracteres del campo.

Este tipo de lista es "muchos a muchos" y tiene el modelo Personas. Permite listar todos los personajes de una película incluidos en la tabla Personas.

Los accesorios necesarios para completar los modelos están incluidos en el archivo ex10_initial_data.json proporcionado con los recursos de hoy.

En esta aplicación, también creará una vista accesible a través de la siguiente URL: 127.0.0.1:8000/ex10. Debe mostrar un formulario con estos campos obligatorios:

- Fecha mínima de estreno de las películas: fecha
- Fecha máxima de lanzamiento de películas: fecha
- Diámetro del planeta mayor que: número

Género del personaje: lista desplegable que muestra los diferentes valores disponibles en el campo de género del modelo Personas. El mismo valor no debe aparecer dos veces.

Una vez validada, la vista debe buscar, devolver y mostrar los resultados.

Un resultado es un personaje cuyo género coincide con el campo 'género del personaje', con una película en la que participa, si la película se estrenó entre la fecha de estreno mínima de Películas y la fecha de estreno máxima de Películas, su planeta si su diámetro es mayor o igual al diámetro del planeta mayor que.

Si su investigación no arroja resultados, debe aparecer el mensaje "Nada que corresponda a su investigación". Cada resultado debe mostrarse en una línea con estos elementos (no necesariamente en este orden):

- Nombre del personaje
- Su género
- Título de la película
- Nombre del mundo natal
- Diámetro del mundo natal

Por ejemplo: los resultados para los personajes femeninos cuyas películas se estrenaron entre "1900-01-01" y "2000-01-01" y cuyo mundo natal tiene un diámetro mayor a 11000 son:

- Una nueva esperanza - Leia Organa - mujer - Alderaan - 12500
- La amenaza fantasma - Padmé Amidala - mujer - Naboo - 12120
- El regreso del Jedi - Leia Organa - mujer - Alderaan - 12500
- El regreso del Jedi - Mon Mothma - mujer - Chandrila - 13500
- El Imperio Contraataca - Leia Organa - mujer - Alderaan - 12500



Pueden estar varios personajes en la misma película y un personaje puede

Aparecen en varias películas. Esto se llama relación de muchos a muchos.

En este caso, se debe crear una tabla intermedia entre

Estas tablas. Cada fila de esta tabla intermedia es una (única)

referencia cruzada: la primera referencia a una fila de la tabla de películas.

La segunda referencia a una fila de la tabla de caracteres (o viceversa). Una vez creados los modelos y realizadas las migraciones, podrá ver esta tabla en la consola de PostgreSQL.

Capítulo XIV

Presentación y evaluación por pares

Entrega tu tarea en tu repositorio de Git como de costumbre. Solo se evaluará el trabajo dentro de tus repositorios durante la defensa. No dudes en verificar los nombres de tus carpetas y archivos para asegurarte de que sean correctos.



El proceso de evaluación se realizará en el computador del evaluado.
grupo.