Tarea: Ecto and Catalogs

1. Cambiar el atributo sex en Pet y HealthExpert para que usen enums (Ecto.Enum)

```
 schema "pets" do
   field :age, :integer
   field :name, :string
   field :sex, Ecto.Enum, values: [:male, :female]
   field :type, :string

   belongs_to :owner, PetClinic.PetOwner.Owner, on_replace: :nilify
   belongs_to :prefferred_expert, PetClinic.PetHealthExpert.Expert,
on_replace: :nilify

   timestamps()
 end
```

```
schema "experts" do
   field :age, :integer
   field :email, :string
   field :name, :string
   field :sex, Ecto.Enum, values: [:male, :female]
   field :specialities, :string

   has_many :patients, PetClinic.PetClinicService.Pet, foreign_key:
:prefferred_expert_id, on_replace: :nilify

   timestamps()
 end
```

2. Crear una migración para corregir el sexo en Pet y HealthExpert, usar un default para cuando no se sabe bien el sexo.

Antes de la migración me dan esos resultados, para Expert si me imprime un resultado porque los sexos que estaban ya registrados estaban correctos, pero para el caso de los Pets ahí no es la misma situación porque adrede puse el sexo de “fred” mal para ver que me marcaba. Entonces es por eso que es necesaria la migración

iex(37)> Repo.all(from p in Pet, select: [p.name, p.sex])

[debug] QUERY OK source="pets" db=1.7ms queue=0.1ms idle=1290.5ms
SELECT p0."name", p0."sex" FROM "pets" AS p0 []
[
  ["clifford", "male"],
  ["yuka", "female"],

```
  ["rocky", "male"],
  ["df", "female"],
  ["Stormy", "female"],
  ["bicho", "female"],
  ["fred", "Malee"]
]
```

iex(42)> Repo.all(from e in Expert, select: [e.name, e.sex])
[debug] QUERY OK source="experts" db=1.0ms queue=1.1ms idle=1393.6ms
SELECT e0."name", e0."sex" FROM "experts" AS e0 []
[["Adrian", :male], ["Regina", :female], ["Amir", :male], ["Erick", :male]]

iex(43)> Repo.all(from p in Pet, select: [p.name, p.sex])
[debug] QUERY OK source="pets" db=1.2ms idle=1939.2ms
SELECT p0."name", p0."sex" FROM "pets" AS p0 []
** (ArgumentError) cannot load `"Malee"` as type {:parameterized, Ecto.Enum, %{mappings: [male: "male", female: "female"], on_cast: %{"female" => :female, "male" => :male}, on_dump: %{female: "female", male: "male"}, on_load: %{"female" => :female, "male" => :male}, type: :string}}
    (ecto 3.7.2) lib/ecto/repo/queryable.ex:409: Ecto.Repo.Queryable.process/4
    (elixir 1.13.3) lib/enum.ex:1715: Enum."-map_reduce/3-lists^mapfoldl/2-0-"/3
    (elixir 1.13.3) lib/enum.ex:1715: Enum."-map_reduce/3-lists^mapfoldl/2-0-"/3
    (ecto 3.7.2) lib/ecto/repo/queryable.ex:273: anonymous fn/3 in Ecto.Repo.Queryable.postprocessor/4
    (elixir 1.13.3) lib/enum.ex:1593: Enum."-map/2-lists^map/1-0-"/2
    (elixir 1.13.3) lib/enum.ex:1593: Enum."-map/2-lists^map/1-0-"/2
    (ecto 3.7.2) lib/ecto/repo/queryable.ex:225: Ecto.Repo.Queryable.execute/4
    (ecto 3.7.2) lib/ecto/repo/queryable.ex:19: Ecto.Repo.Queryable.all/3

```
defmodule PetClinic.Repo.Migrations.CorrectSex do
 use Ecto.Migration
 alias PetClinic.Repo

 def change do
   queryP = "update pets set sex = lower(sex)"
   Ecto.Adapters.SQL.query!(Repo, queryP, [])

   queryP = "update pets set sex = 'female' where sex not in ('female', 'male')"
   Ecto.Adapters.SQL.query!(Repo, queryP, [])

   queryE = "update experts set sex = lower(sex)"
   Ecto.Adapters.SQL.query!(Repo, queryE, [])

   queryE = "update experts set sex = 'female' where sex not in ('female', 'male')"
```

```
    Ecto.Adapters.SQL.query!(Repo, queryE, [])
  end
end
```

RESULTADO AL EJECUTAR LA MIGRACIÓN
mix ecto.migrate
Compiling 2 files (.ex)

17:25:32.113 [info] == Running 20220422175850
PetClinic.Repo.Migrations.CorrectSex.change/0 forward

17:25:32.153 [debug] QUERY OK db=1.7ms
update pets set sex = lower(sex) []

17:25:32.155 [debug] QUERY OK db=1.4ms
update pets set sex = 'female' where sex not in ('female', 'male') []

17:25:32.156 [debug] QUERY OK db=0.6ms
update experts set sex = lower(sex) []

17:25:32.157 [debug] QUERY OK db=0.3ms
update experts set sex = 'female' where sex not in ('female', 'male') []

17:25:32.161 [info] == Migrated 20220422175850 in 0.0s

RESULTADO DESPUÉS DE LA MIGRACIÓN AL CONSULTAR PETS
iez(44)> Repo.all(from p in Pet, select: [p.name, p.sex])
[debug] QUERY OK source="pets" db=0.7ms queue=0.1ms idle=1878.4ms
SELECT p0."name", p0."sex" FROM "pets" AS p0 []
[
  ["clifford", :male],
  ["yuka", :female],
  ["rocky", :male],
  ["df", :female],
  ["Stormy", :female],
  ["bicho", :female],
  ["fred", :female]
]
iex(45)> Repo.all(from e in Expert, select: [e.name, e.sex])
[debug] QUERY OK source="experts" db=3.0ms queue=0.1ms idle=1433.2ms
SELECT e0."name", e0."sex" FROM "experts" AS e0 []
[["Adrian", :male], ["Regina", :female], ["Amir", :male], ["Erick", :male]]
iex(46)>

3. Cambiar Pet para que type no sea un string, sino un catálogo PetType. Esto implica:

- Crear el schema de PetType

```
defmodule PetClinic.PetType do
 use Ecto.Schema
 import Ecto.Changeset

 schema "pet_types" do
   field :name, :string

   timestamps()
 end
end
```

- Crear la migración para crear la tabla, cambiar la relación de pet y migrar los datos.

```
defmodule PetClinic.Repo.Migrations.CreatePetTypes do
 use Ecto.Migration
 alias PetClinic.Repo
 alias PetClinic.PetClinicService.Pet
 alias PetClinic.PetType
 import Ecto.Query

 def change do
   create table(:pet_types) do
     add :name, :string

     timestamps()
   end

   query = "select id, type from pets"
   pets = Ecto.Adapters.SQL.query!(Repo, query, []) |> Map.get(:rows)

   query = "select distinct type from pets"
   types = Ecto.Adapters.SQL.query!(Repo, query, []) |> Map.get(:rows)
|> List.flatten()


   flush()

   Enum.each(types, fn t ->
```

```
    Repo.insert(%PetType{name: t})
  end)

  alter table("pets") do
    remove :type
    add :type_id, references("pet_types")
  end

  flush()

  IO.inspect(pets);

  Enum.each(pets, fn [pet_id, pet_type] ->
    %PetType{id: pet_type_id} = Repo.get_by(PetType, name: pet_type)
    update = "update pets set type_id = $1::integer where id =
$2::integer"
    Ecto.Adapters.SQL.query!(Repo, update, [pet_type_id, pet_id])
  end)

 end
end
```

ANTES DE LA MIGRACIÓN

ex(9)> Repo.all(Pet)

[debug] QUERY OK source="pets" db=0.7ms queue=0.2ms idle=1804.7ms
SELECT p0."id", p0."age", p0."name", p0."sex", p0."type", p0."owner_id", p0."expert_id", p0."inserted_at", p0."updated_at" FROM "pets" AS p0 []
[
  %PetClinic.PetClinicService.Pet{
    __meta__: #Ecto.Schema.Metadata<:loaded, "pets">,
    age: 3,
    expert: #Ecto.Association.NotLoaded<association :expert is not loaded>,
    expert_id: nil,
    id: 1,
    inserted_at: ~N[2022-04-25 01:23:16],
    name: "bicho",
    owner: #Ecto.Association.NotLoaded<association :owner is not loaded>,
    owner_id: nil,
    sex: :male,
    type: "dog",
    updated_at: ~N[2022-04-25 01:23:16]

```
  },
  %PetClinic.PetClinicService.Pet{
        __meta__: #Ecto.Schema.Metadata<:loaded, "pets">,
        age: 4,
        expert: #Ecto.Association.NotLoaded<association :expert is not loaded>,
        expert_id: nil,
        id: 2,
        inserted_at: ~N[2022-04-25 01:26:22],
        name: "yuka",
        owner: #Ecto.Association.NotLoaded<association :owner is not loaded>,
        owner_id: nil,
        sex: :female,
        type: "cat",
        updated_at: ~N[2022-04-25 01:26:22]
  },
  %PetClinic.PetClinicService.Pet{
        __meta__: #Ecto.Schema.Metadata<:loaded, "pets">,
        age: 6,
        expert: #Ecto.Association.NotLoaded<association :expert is not loaded>,
        expert_id: nil,
        id: 3,
        inserted_at: ~N[2022-04-25 01:26:43],
        name: "fred",
        owner: #Ecto.Association.NotLoaded<association :owner is not loaded>,
        owner_id: nil,
        sex: :male,
        type: "snake",
        updated_at: ~N[2022-04-25 01:26:43]
  }
]
```

DESPUÉS DE LA MIGRACIÓN

ex(21)> Repo.all(Pet) |> Repo.preload(:type)

```
[debug] QUERY OK source="pets" db=2.0ms queue=0.1ms idle=1545.0ms
SELECT p0."id", p0."age", p0."name", p0."sex", p0."type_id", p0."owner_id",
p0."expert_id", p0."inserted_at", p0."updated_at" FROM "pets" AS p0 []
[debug] QUERY OK source="pet_types" db=1.1ms queue=1.4ms idle=1554.7ms
SELECT p0."id", p0."name", p0."inserted_at", p0."updated_at", p0."id" FROM
"pet_types" AS p0 WHERE (p0."id" = ANY($1)) [[3, 1, 2]]
[
  %PetClinic.PetClinicService.Pet{
        __meta__: #Ecto.Schema.Metadata<:loaded, "pets">,
        age: 3,
```

```
        expert: #Ecto.Association.NotLoaded<association :expert is not loaded>,
        expert_id: nil,
        id: 1,
        inserted_at: ~N[2022-04-25 01:23:16],
        name: "bicho",
        owner: #Ecto.Association.NotLoaded<association :owner is not loaded>,
        owner_id: nil,
        sex: :male,
        type: %PetClinic.PetType{
        __meta__: #Ecto.Schema.Metadata<:loaded, "pet_types">,
        id: 2,
        inserted_at: ~N[2022-04-25 16:40:42],
        name: "dog",
        updated_at: ~N[2022-04-25 16:40:42]
        },
        type_id: 2,
        updated_at: ~N[2022-04-25 01:23:16]
},
%PetClinic.PetClinicService.Pet{
        __meta__: #Ecto.Schema.Metadata<:loaded, "pets">,
        age: 4,
        expert: #Ecto.Association.NotLoaded<association :expert is not loaded>,
        expert_id: nil,
        id: 2,
        inserted_at: ~N[2022-04-25 01:26:22],
        name: "yuka",
        owner: #Ecto.Association.NotLoaded<association :owner is not loaded>,
        owner_id: nil,
        sex: :female,
        type: %PetClinic.PetType{
        __meta__: #Ecto.Schema.Metadata<:loaded, "pet_types">,
        id: 1,
        inserted_at: ~N[2022-04-25 16:40:42],
        name: "cat",
        updated_at: ~N[2022-04-25 16:40:42]
        },
        type_id: 1,
        updated_at: ~N[2022-04-25 01:26:22]
},
%PetClinic.PetClinicService.Pet{
        __meta__: #Ecto.Schema.Metadata<:loaded, "pets">,
        age: 6,
        expert: #Ecto.Association.NotLoaded<association :expert is not loaded>,
        expert_id: nil,
```

```
        id: 3,
        inserted_at: ~N[2022-04-25 01:26:43],
        name: "fred",
        owner: #Ecto.Association.NotLoaded<association :owner is not loaded>,
        owner_id: nil,
        sex: :male,
        type: %PetClinic.PetType{
        __meta__: #Ecto.Schema.Metadata<:loaded, "pet_types">,
        id: 3,
        inserted_at: ~N[2022-04-25 16:40:42],
        name: "snake",
        updated_at: ~N[2022-04-25 16:40:42]
        },
        type_id: 3,
        updated_at: ~N[2022-04-25 01:26:43]
  }
]
```

Enviar:
- Link al repo
- Link a c/u de las migraciones
- Evidencia del iex del antes y después de la migración.
Ejemplo:
- Antes: iex> Repo.all(Pet)
- Después: iex> Repo.all(Pet) |> Repo.preload(:type)