Trabajo Tutelado

MAI 24/25

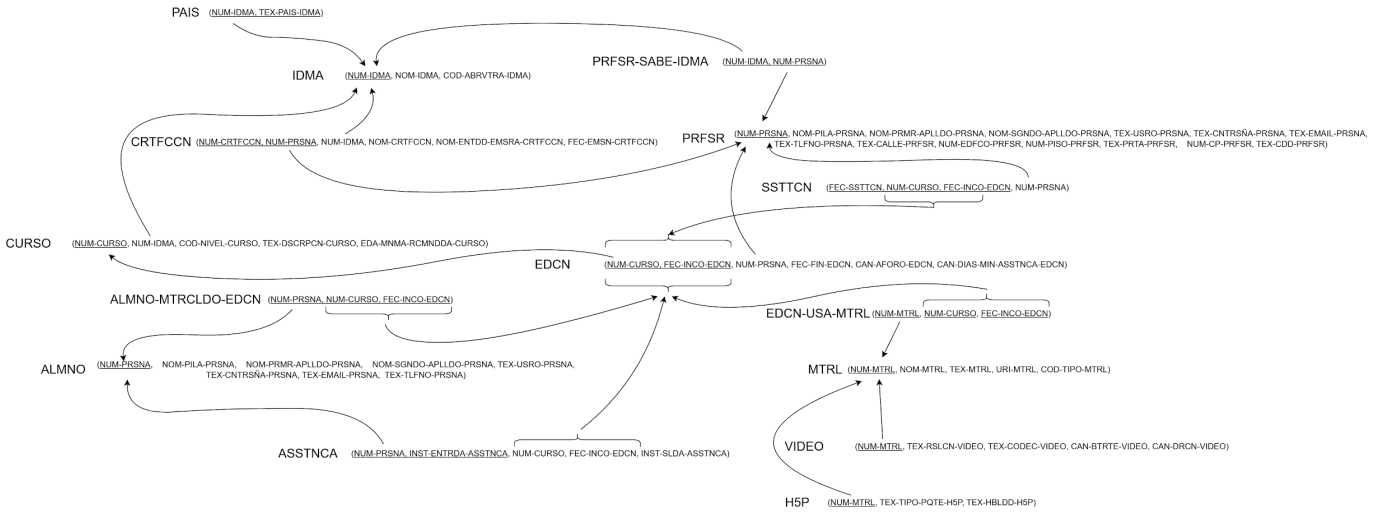
Adrián Edreira Gantes
adrian.gantes@udc.es

Lucas García García
lucas.garcia.garcia@udc.es

1. Modelo Relacional

Esquema

Se adjunta también la siguiente imagen con el nombre “Modelo_Relacional.png” para mayor claridad y legibilidad.



Justificaciones

La mayor parte de este diseño del modelo relacional sigue las normas establecidas para este modelo. No obstante, existen dos decisiones de diseño que se comentan a continuación.

Se ha decidido utilizar una tabla para cada subclase de persona, de esta forma el identificador será el heredado de la superclase “Persona” y compartirán los atributos heredados y mantendrán los propios de cada subclase, así como sus relaciones. Se ha decidido usar este diseño debido a las relaciones de las subclases con el resto de entidades y la falta de relaciones de la superclase “Persona”.

Respecto a la entidad “Material” y sus subclases se ha utilizado una tabla para cada entidad. Cada tabla contiene los atributos propios de su entidad, la superclase contiene el identificador como clave primaria (“NUM-MTRL”) y las subclases tienen dicha clave primaria que es a su vez una clave foránea a la tabla de la superclase. Se ha decidido usar este diseño ya que se pueden introducir nuevos tipos de materiales en el futuro a la base de datos y las subclases no tienen relaciones propias.

2. ODL

Esquema

Se adjunta también un documento *txt* llamado “Esquema_ODL.txt” para facilitar la corrección.

```
Class Material (extent Materiales){

    attribute integer id_material;
    attribute string nombre;
    attribute string tipo_recurso;
    attribute string url;
    attribute string descripción;


    relationship set<Edición> ediciones
      inverse Edición::materiales;
}
```

```
Class Sustitución (extent Sustituciones){

    attribute date fecha;


    relationship Edición edición
            inverse Edición::sustituciones;

    relationship Profesor profesor
            inverse Profesor::sustituciones;
}
```

```
Class Persona (extent Personas){

    attribute integer id_persona;
    attribute string email;
    attribute string teléfono;
    attribute string usuario;
    attribute string contraseña;
    attribute struct{
        string pila;
        string apellido1;
        string apellido2;
    }nombre;
}
```

```
Class Asistencia (extent Asistencias){

    attribute dateTime instante_entrada;
    attribute dateTime instante_salida;


    relationship Alumno alumno
            inverse Alumno::asistencias;

    relationship Edición edición
            inverse Edición::asistencias;
}
```

```
Class Certificación (extent Certificaciones){

    attribute integer id_certificación;
    attribute string nombre;
    attribute string entidad_emisora;
    attribute date fecha_emisión;


    relationship Profesor profesor
            inverse Profesor::certificaciones;

    relationship Idioma idioma
            inverse Idioma::certificaciones;
}
```

```
Class Curso (extent Cursos){

    attribute integer id_curso;
    attribute string nivel;
    attribute string descripción;
    attribute integer edad_mínima;


    relationship Idioma idioma
            inverse Idioma::cursos;

    relationship set<Edición> ediciones
            inverse Edición::curso;
}
```

```
Class Idioma (extent Idiomas){

    attribute integer id_idioma;
    attribute string nombre;
    attribute string abreviatura;
    attribute set<string> país;


    relationship set<Certificación> certificaciones
            inverse Certificación::idioma;

    relationship set<Profesor> profesores
            inverse Profesor::idiomas;

    relationship set<Curso> cursos
            inverse Curso::idioma;
}
```

```
Class Alumno: Persona (extent Alumnos){

    relationship set<Edición> ediciones
            inverse Edición::alumnos;

    relationship set<Asistencia> asistencias
            inverse Alumno::asistencia;
}
```

```
Class Vídeo: Material (extent Vídeos){

    attribute integer duración;
    attribute integer bitrate;
    attribute string codec;
    attribute string resolución;
}
```

```
Class H5P: Material (extent H5Ps){

    attribute string habilidad;
    attribute string tipo_paquete;
}
```

```
Class Profesor: Persona (extent Profesores){

    attribute struct{
        string calle;
        integer número;
        string piso;
        string puerta;
        integer cp;
        string ciudad;
    }dirección;


    relationship set<Certificación> certificaciones
            inverse Certificación::profesor;

    relationship set<Idioma> idiomas
            inverse Idioma::profesores;

    relationship set<Sustitución> sustituciones
            inverse Sustitución::profesor;

    relationship set<Edición> ediciones
            inverse Edición::profesor;
}
```

```
Class Edición (extent Ediciones){

    attribute date fecha_inicio;
    attribute date fecha_fin;
    attribute integer aforo;
    attribute integer días_mínimo_asistencia;


    relationship set<Material> materiales
            inverse Material::ediciones;

    relationship set<Alumno> alumnos
            inverse Alumno::ediciones;

    relationship set<Asistencia> asistencias
            inverse Asistencia::edición;

    relationship Profesor profesor
            inverse Profesor::ediciones;

    relationship Set<Sustitución> sustituciones
            inverse Sustitución::edición;

    relationship Curso curso
            inverse Curso::ediciones;
}
```

Justificaciones

Se ha creado una clase para cada tipo de entidad y se le han añadido sus correspondientes relaciones, los atributos compuestos se han creado con un Struct y los multivaluados con un Set. Para las relaciones N:N o N:1 se ha usado un Set en las entidades correspondientes. Para las herencias se ha utilizado el modelo “hijo:padre”. Consideramos que no son necesarias más justificaciones ya que se han seguido las normas especificadas para los esquemas ODL.

3. MongoDB

Patrones de acceso

- Gran cantidad de consultas a Persona para el inicio de sesión y cargar el perfil según el tipo de persona. Para un profesor se incluirán los idiomas que sabe. Para un alumno se incluirán las ediciones en las que está matriculado y el total de asistencias a dichas ediciones. Inserción y modificación media con borrados inexistentes.

- Se consultan las listas de idiomas que incluyen información de sus cursos disponibles, el id de dicho curso debe ser único en todo el documento, como se representa en el EER. Modificaciones y borrados prácticamente inexistentes.

- No se consulta con mucha frecuencia la tabla de asistencias, pero es necesaria para registrar la trazabilidad y el número total de asistencias a una edición por un alumno. No tiene modificaciones ni borrados, pero sí gran cantidad de inserciones.

- Consultas frecuentes por usuarios para inscribirse a ediciones abiertas y con plazas libres. Consultas muy frecuentes a ediciones activas por sus alumnos y profesores. Cerca de 100 nuevas ediciones cada año. La edición contiene el número de plazas libres y el nombre completo del profesor.

Documentos

Idioma:

```
practica> db.idiomas.findOne()
{
  _id: 1,
  nombre: 'Español',
  abreviatura: 'ES',
  paises: [ 'España', 'Colombia', 'Venezuela' ],
  cursos: [
    {
      id_curso: 1,
      nivel: 'C2',
      descripcion: 'Español más avanzado, nivel C2',
      edad_minima: 18
    },
    {
      id_curso: 2,
      nivel: 'A1',
      descripcion: 'Español básico',
      edad_minima: 5
    }
  ]
}
```

Asistencia:

```
practica> db.asistencias.findOne()
{
  _id: ObjectId('6830a075046ec3383bb71236'),
  id_persona: 2,
  id_edicion: ObjectId('68303cd6353e25cb3a262089'),
  instante_entrada: ISODate('2025-06-21T16:44:00.000Z'),
  instante_salida: ISODate('2025-06-21T17:32:00.000Z')
}
```

Edición:

```
practica> db.edicion.findOne()
{
  _id: ObjectId('68303cd6353e25cb3a262089'),
  id_curso: 1,
  fecha_inicio: ISODate('2025-06-01T00:00:00.000Z'),
  fecha_fin: ISODate('2025-08-30T00:00:00.000Z'),
  aforo: 10,
  plazas_libres: 2,
  dias_min_asistencia: 3,
  profesor: {
    id_profesor: 1,
    nombre: { pila: 'Juan', apellido1: 'Ríos', apellido2: 'Mate' }
  }
}
```

Persona (Profesor):

```
practica> db.personas.findOne({tipo: 'P'})
{
  _id: 1,
  email: 'juan.rios@academy.com',
  telefono: '+34709985435',
  usuario: 'juan.rios',
  contrasena: 'u#!4OkRP)D',
  nombre: { pila: 'Juan', apellido1: 'Ríos', apellido2: 'Mate' },
  tipo: 'P',
  profesor: {
    direccion: {
      ciudad: 'Sevilla',
      cp: 12411,
      puerta: 'C',
      piso: 4,
      numero: '47',
      calle: 'Callejón Lope Galvez'
    },
    idiomas: [
      { id_idioma: 1, nombre: 'Español' },
      { id_idioma: 2, nombre: 'Inglés' }
    ]
  }
}
```

Persona (Alumno):

```
practica> db.personas.findOne({tipo: 'A'})
{
  _id: 2,
  email: 'flavia.ayuso@academy.com',
  telefono: '+34723802620',
  usuario: 'flavia.ayuso',
  contrasena: 'A+u4BaQr)P',
  nombre: { pila: 'Flavia', apellido1: 'Ayuso', apellido2: 'Gonzalez' },
  tipo: 'A',
  alumno: {
    matriculas: [
      {
        id_edicion: ObjectId('68303cd6353e25cb3a262089'),
        total_asistencias: 5
      },
      {
        id_edicion: ObjectId('68303cd6353e25cb3a26209c'),
        total_asistencias: 0
      }
    ]
  }
}
```

Justificaciones

Para la creación de documentos se ha creado el esquema a seguir y posteriormente se ha utilizado ChatGPT únicamente para crear los archivos con información suficiente para importarlos a MongoDB y por último realizar las consultas correspondientes.

En relación a los patrones de acceso, se ha considerado que lo más utilizado respecto a los usuarios sería iniciar sesión en la plataforma, por eso se han agrupado todos en el mismo conjunto. Se ha considerado que las ediciones se consultarán por separado, buscando ediciones abiertas y con disponibilidad para inscribirse. Los idiomas muestran también los cursos disponibles en la academia para que se puedan buscar ediciones correspondientes a dichos cursos. Por último, las asistencias no serán prácticamente consultadas, por eso no se registra más información de la necesaria, estando separadas para poder realizar gran cantidad de inserciones.

En cuanto a cada documento, en edición disponemos de “id_curso” como referencia a “idioma.cursos.id_curso”, de esta forma podemos obtener toda la información de una edición si fuese necesario. Además tenemos un atributo calculado (computed), “plazas_libres”, para saber el número de plazas disponibles para cierta edición, sin necesidad de calcularlo mediante consultas. También guardamos la información del profesor haciendo uso de referencia extendida para representar la información siguiendo el patrón de acceso.

Lo que destaca en el documento de idiomas es el atributo “cursos.id_curso”, ya que debe ser único en todo el conjunto de idiomas, es decir, no puede haber dos cursos con el mismo id, aunque sean de diferentes idiomas.

Para las asistencias creamos un documento nuevo para cada asistencia, de esta forma no tendremos un gran array por cada edición de un alumno. Con una simple agrupación podremos obtener las asistencias correspondientes. Además, tenemos referencias tanto al conjunto de personas como al de ediciones por si fuese necesario obtener más información en algún momento puntual.

Por último, cada persona tiene su correspondiente información y dependiendo del tipo tendrán una u otra. En el caso del profesor los idiomas que sabe haciendo uso de referencia con el id y referencia extendida con el nombre del idioma. En el caso del alumno, se hace uso de referencia hacia el id de una edición y el atributo calculado “total_asistencias” para saber el número de asistencias correspondientes a cierta edición y saber si es apto para recibir el certificado.

Consultas

Obtener todos las ediciones disponibles que se han iniciado este año con espacio para inscripciones nuevas:

```
db.ediciones.find({plazas_libres: { $gt: 0 }, fecha_inicio:{$gte: new Date("2025-01-01")}})
```

Idiomas que solo saben dos profesores o menos:

```
db.personas.aggregate([
  { $match: { tipo: 'P' } },
  { $unwind: "$profesor.idiomas" },
  { $group: {
    _id: "$profesor.idiomas.id_idioma",
    nombre: { $first: "$profesor.idiomas.nombre" },
    total_profesores: { $sum: 1 }
  }},
  { $match: { total_profesores: { $lte: 2 } } },
  { $project: { _id: 0, id_idioma: "$_id", nombre: 1, total_profesores: 1 } }
])
```

Todos los cursos diferentes que ha impartido cada profesor con su nombre completo:

```
db.ediciones.aggregate([
 {
  $group: {
   _id: "$profesor.id_profesor",
   nombre: {$first: "$profesor.nombre.pila"},
   apellido1: {$first: "$profesor.nombre.apellido1"},
   apellido2: {$first: "$profesor.nombre.apellido2"},
   cursos_distintos: { $addToSet: "$id_curso" }
  }
 },
 {
  $project: {
   _id: 0,
   id_profesor: "$_id",
   nombre: "$nombre",
   apellido1: "$apellido1",
   apellido2: "$apellido2",
   cursos_distintos: 1
  }
 }
])
```

Todos los idiomas que hayan tenido alguna edición que imparta un curso con el nivel C2:

```
db.idiomas.aggregate([
 { $unwind: "$cursos" },
 { $match: { "cursos.nivel": "C2" } },
 {
  $lookup: {
  from: "ediciones",
  localField: "cursos.id_curso",
  foreignField: "id_curso",
  as: "ediciones"
  }
 },
 { $match: { "ediciones.0": { $exists: true } } },
 {
  $group: {
  _id: "$_id",
  nombre: { $first: "$nombre" },
  abreviatura: { $first: "$abreviatura" },
  paises: { $first: "$paises" }
  }
 }
])
```

Alumnos que han superado los días mínimos de asistencias en ediciones que estuviesen matriculados:

```
db.personas.aggregate([
 { $match: { tipo: 'A' } },
 { $unwind: "$alumno.matriculas" },
 {
  $lookup: {
  from: "ediciones",
  localField: "alumno.matriculas.id_edicion",
  foreignField: "_id",
  as: "edi"
  }
 },
 { $unwind: "$edi" },
 {
  $match: {
   $expr: {
    $gte: ["$alumno.matriculas.total_asistencias", "$edi.dias_min_asistencia"]
   }
  }
 },
 {
   $group: {
    _id: "$alumno.matriculas.id_edicion",
    usuarios: { $addToSet: "$usuario" }
   }
 },
 {
  $project: {
  _id: 0,
  id_edicion: "$_id",
  usuarios: 1
  }
 }
])
```