



SISTEMAS DE INFORMAÇÃO E BASE DE DADOS

APRESENTAÇÃO

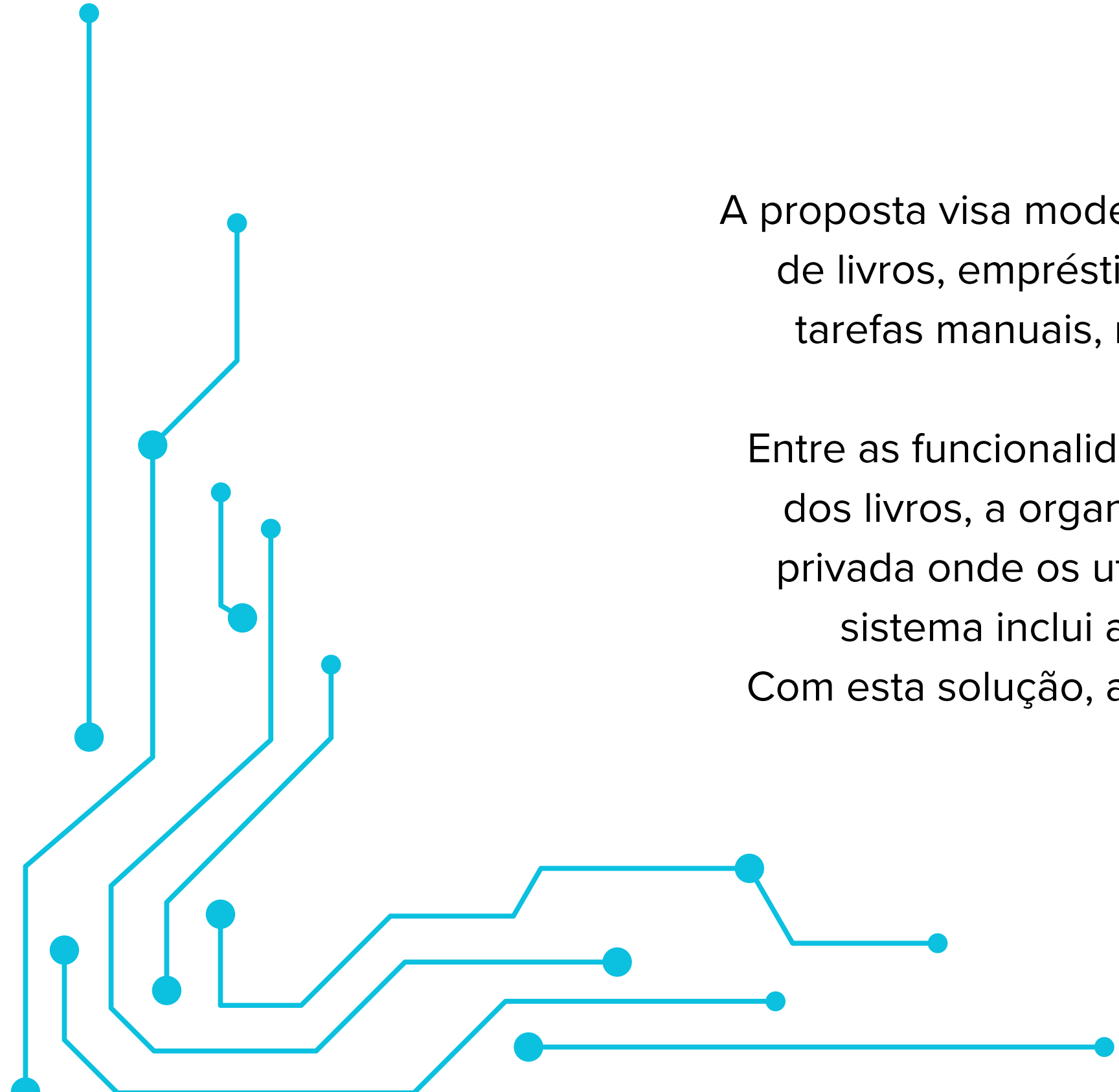
BIBLIOTECA UNIVERSITÁRIA

Catarina Moutinho
Francisca Silva
Marcela Almeida





A NOSSA PROPOSTA



A proposta visa modernizar a gestão da Biblioteca Universitária, otimizando o controlo de livros, empréstimos, devoluções, reservas e penalizações. O sistema automatiza tarefas manuais, reduz o tempo de atendimento e melhora o acesso à informação, beneficiando tanto os funcionários quanto os utilizadores. Entre as funcionalidades, destacam-se o acompanhamento em tempo real do estado dos livros, a organização física detalhada (corredor, estante, prateleira), e uma área privada onde os utilizadores podem reservar, renovar e consultar o seu histórico. O sistema inclui ainda notificações automáticas e geração de relatórios de gestão. Com esta solução, a biblioteca torna-se mais eficiente, organizada e alinhada com as necessidades da comunidade académica atual.


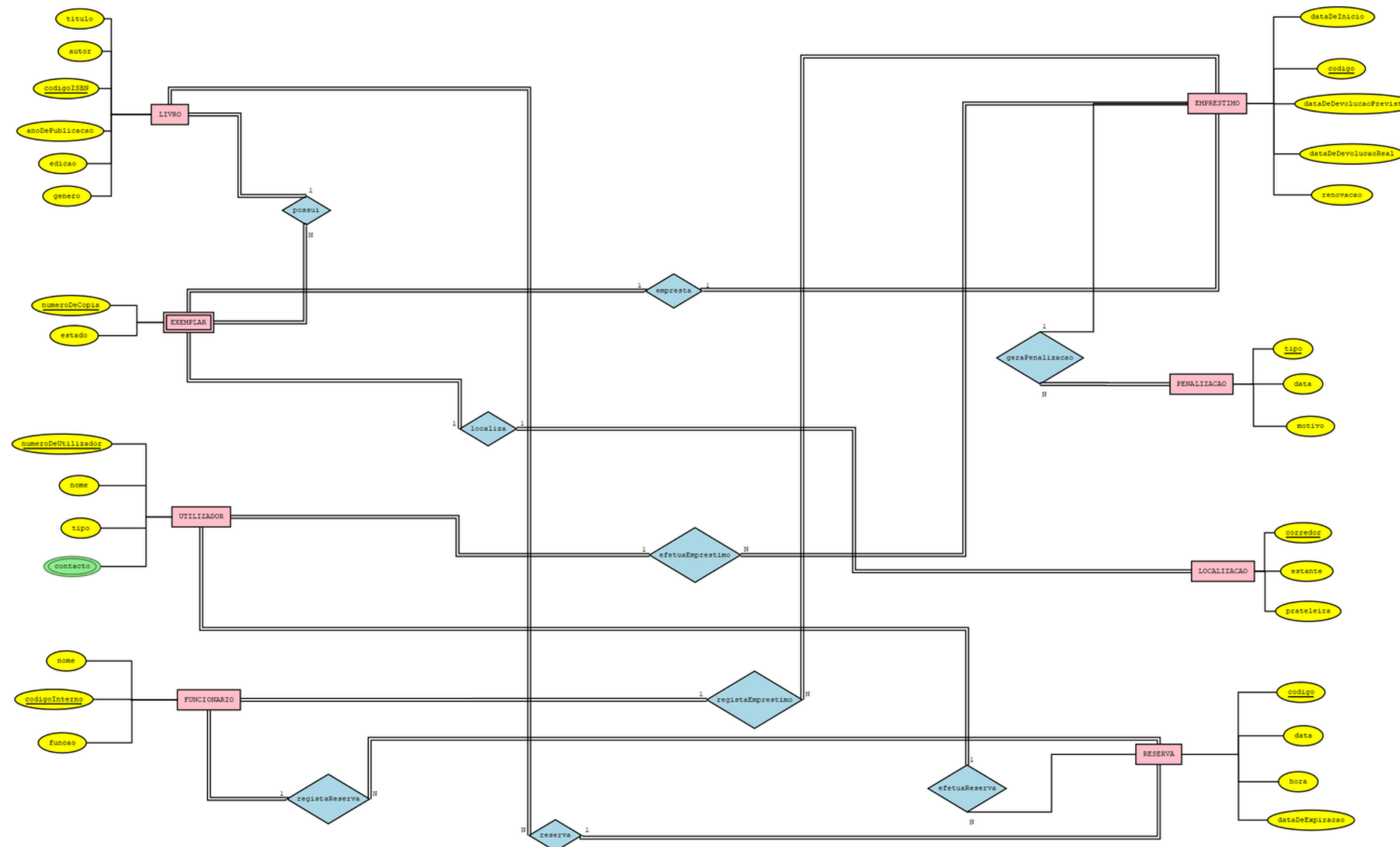


DIAGRAMA ENTIDADE-ASSOCIAÇÃO (ANTIGO)



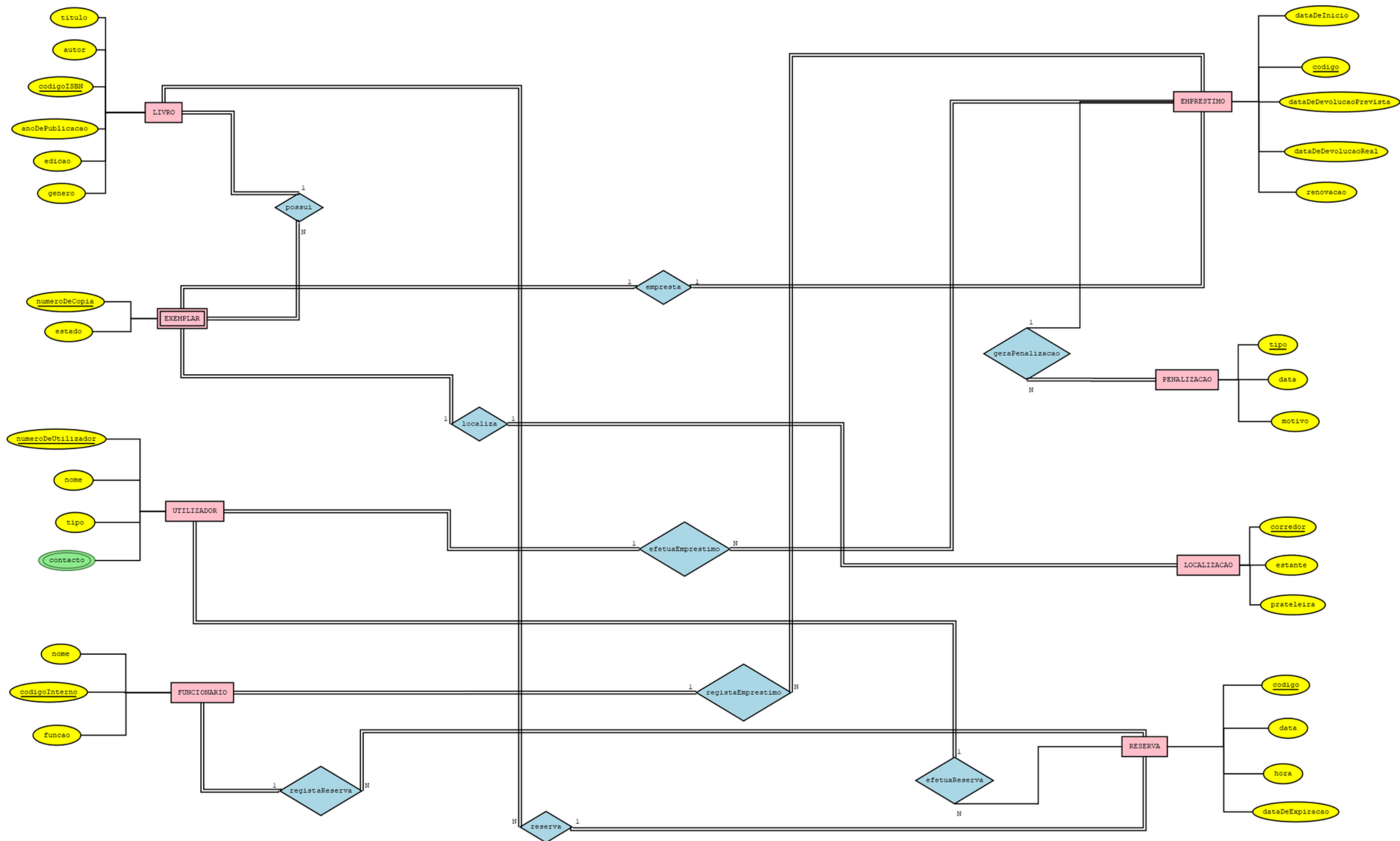
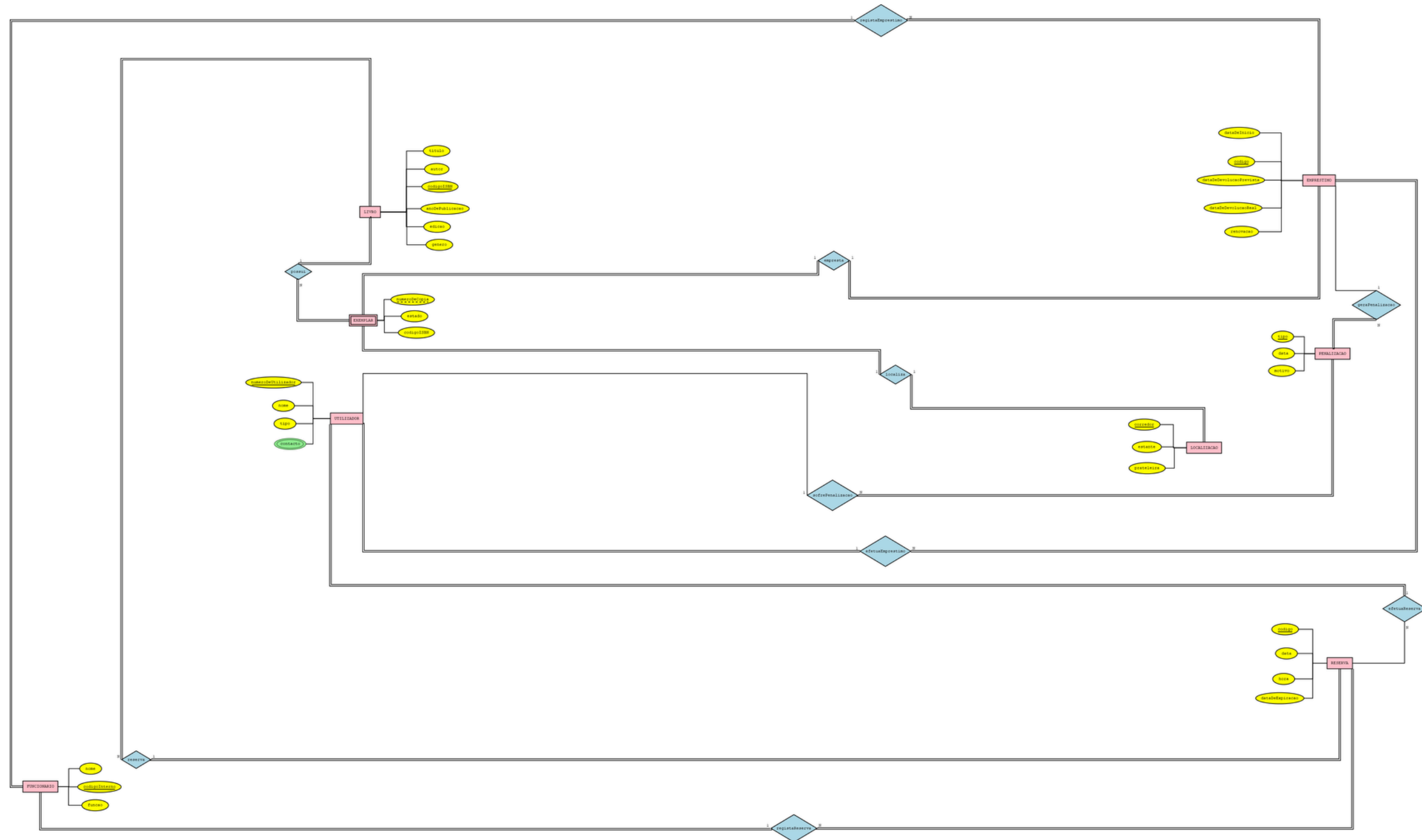
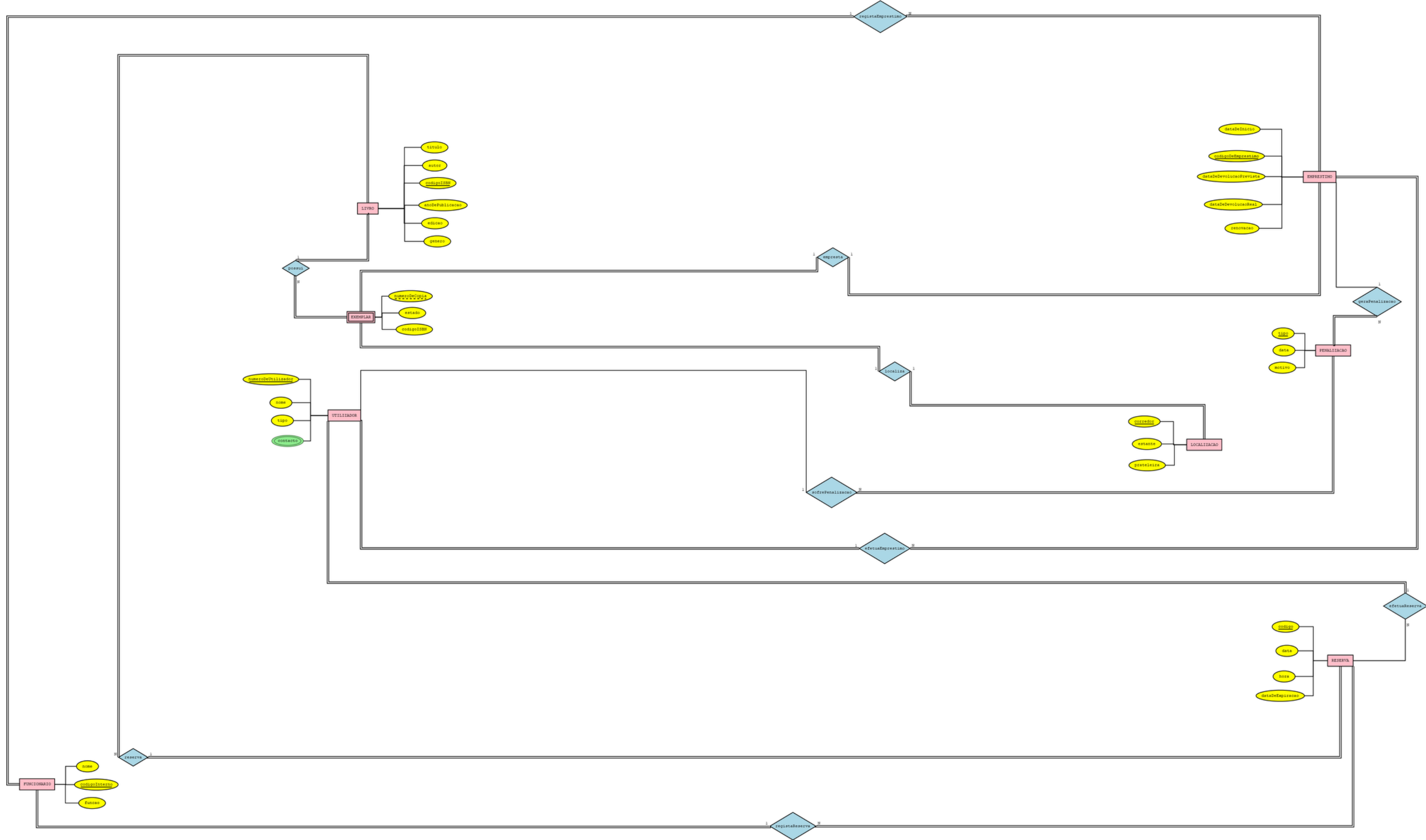


DIAGRAMA ENTIDADE-ASSOCIAÇÃO (FINAL)



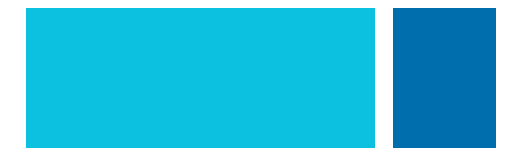
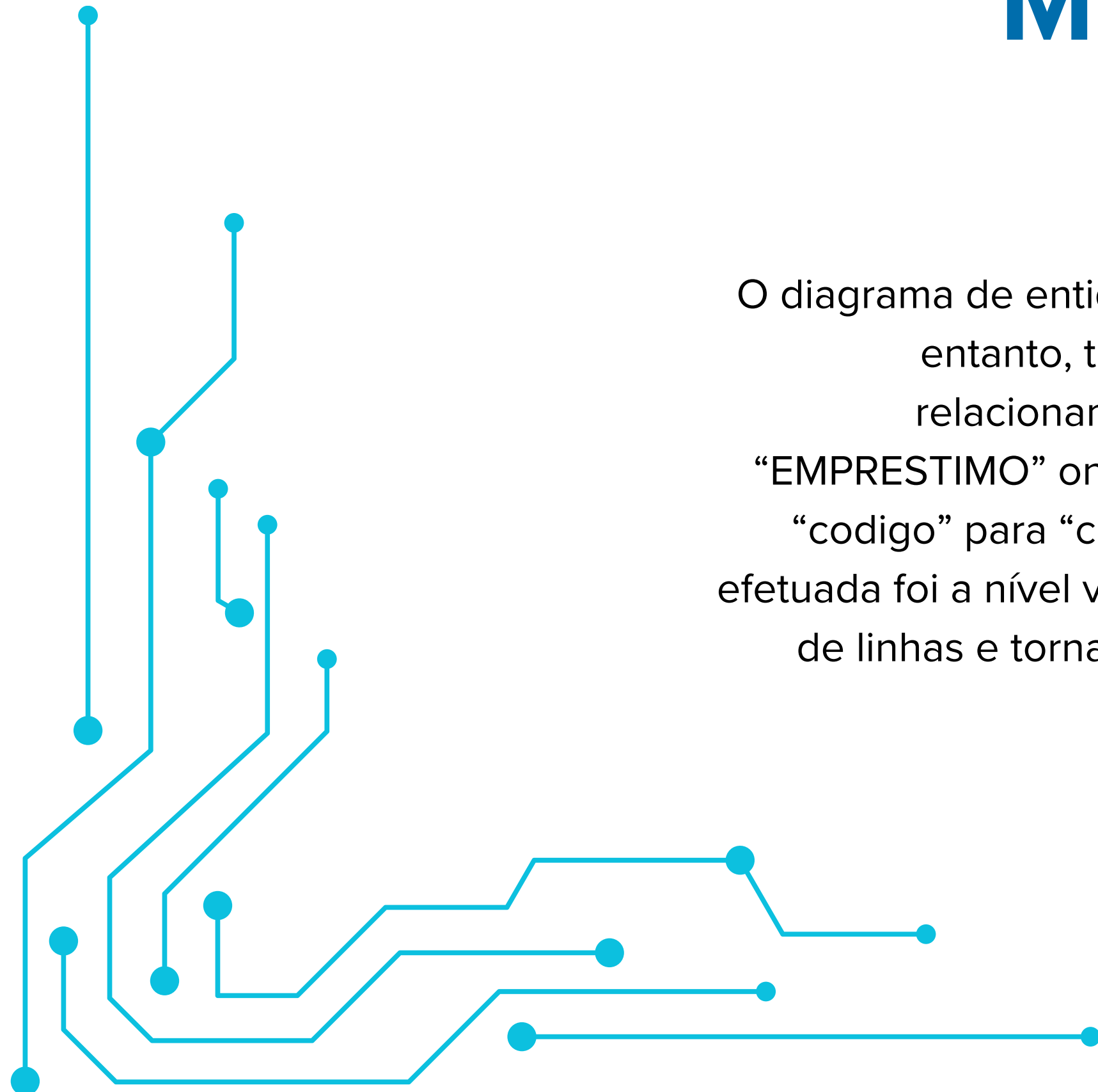




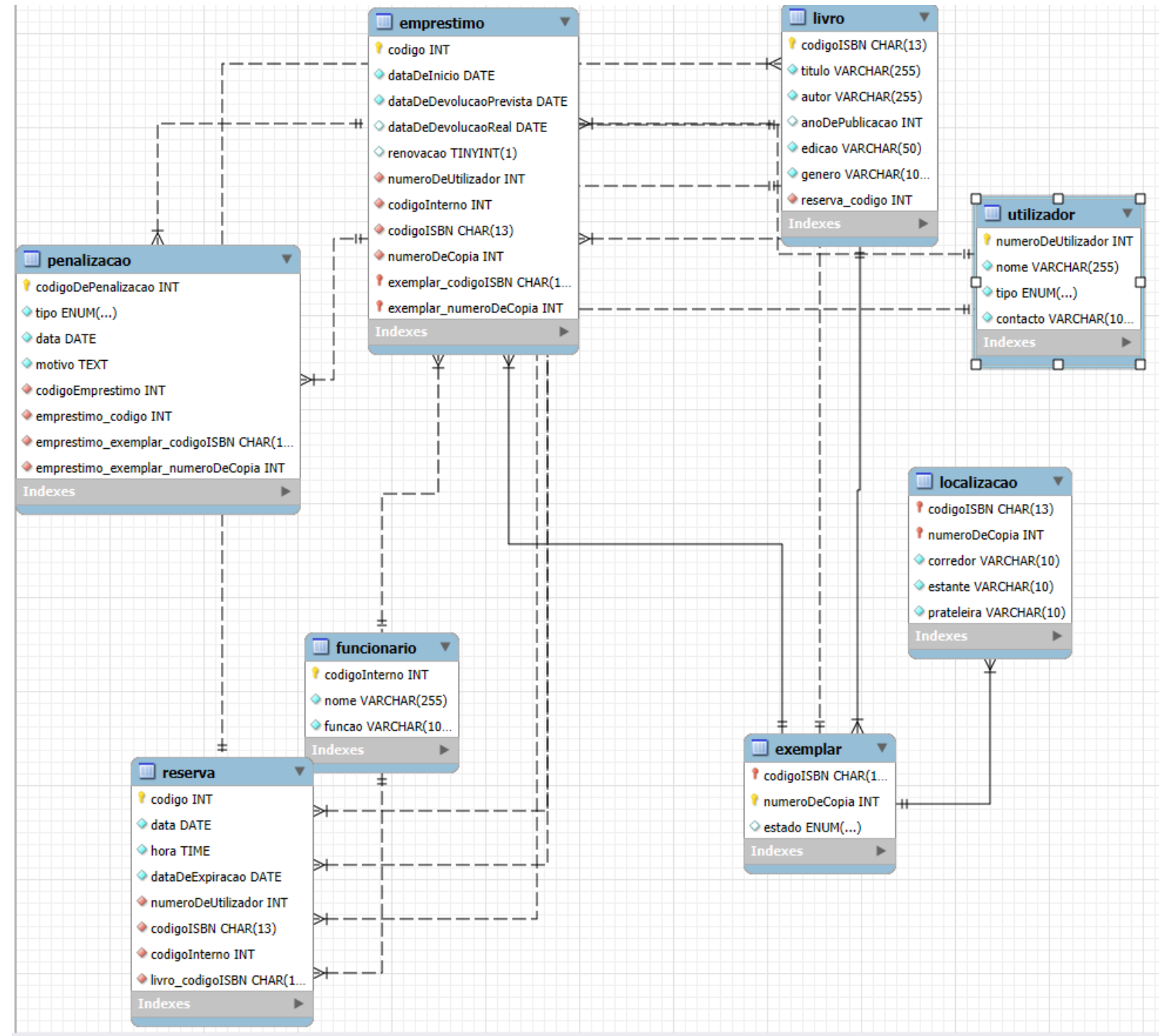
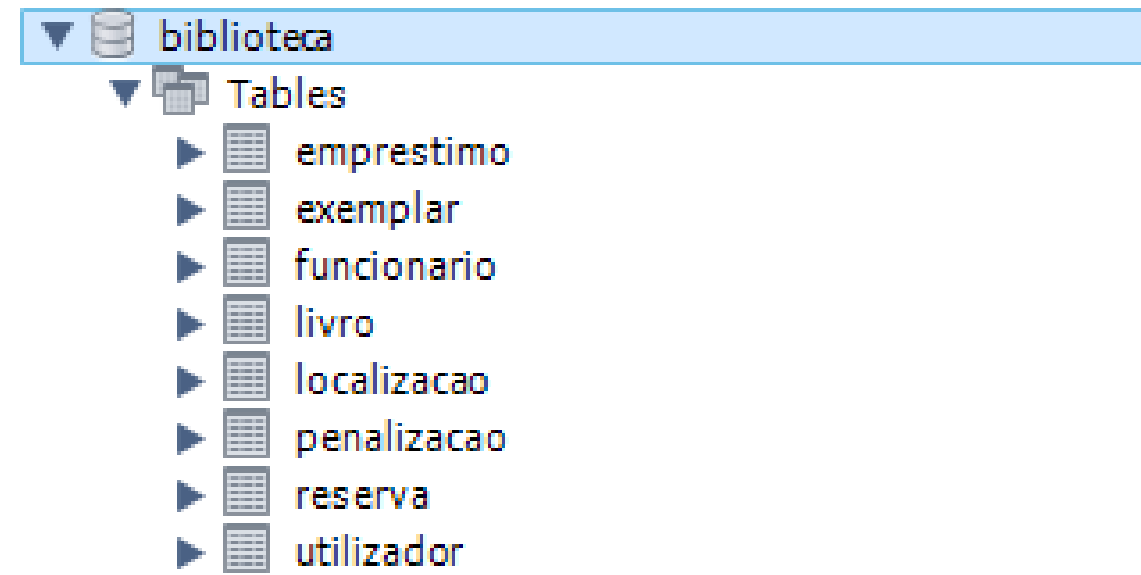
MUDANÇAS REALIZADAS

NO DIAGRAMA ENTIDADE-ASSOCIAÇÃO

O diagrama de entidade-associação foi reorganizado graficamente, mantendo-se, no entanto, toda a estrutura original inalterada. As entidades, os atributos e os relacionamentos permanecem exatamente os mesmos, exceto na entidade “EMPRESTIMO” onde tivemos de fazer uma pequena alteração no nome do atributo “codigo” para “codigoDeEmprestimo”, para assim ficar mais claro. Outra alteração efetuada foi a nível visual, com o objetivo de melhorar a clareza, reduzir o cruzamento de linhas e tornar o diagrama mais limpo e fácil de interpretar. Esta reorganização facilita a leitura e a compreensão do modelo.



DATABASE MODEL



MYSQL - DDL

```
CREATE TABLE IF NOT EXISTS Utilizador (  
  numeroDeUtilizador INT AUTO_INCREMENT PRIMARY KEY,  
  nome VARCHAR(255) NOT NULL,  
  tipo ENUM('Aluno', 'Professor', 'Funcionário') NOT NULL,  
  contacto VARCHAR(100) NOT NULL  
);
```

```
CREATE TABLE IF NOT EXISTS Funcionario (  
  codigoInterno INT AUTO_INCREMENT PRIMARY KEY,  
  nome VARCHAR(255) NOT NULL,  
  funcao VARCHAR(100) NOT NULL  
);
```

```
CREATE TABLE IF NOT EXISTS Livro (  
  codigoISBN CHAR(13) PRIMARY KEY,  
  titulo VARCHAR(255) NOT NULL,  
  autor VARCHAR(255) NOT NULL,  
  anoDePublicacao INT CHECK (anoDePublicacao >= 1500),  
  edicao VARCHAR(50) NOT NULL,  
  genero VARCHAR(100) NOT NULL  
);
```

```
CREATE TABLE IF NOT EXISTS Exemplar (  
  codigoISBN CHAR(13) NOT NULL,  
  numeroDeCopia INT NOT NULL,  
  estado ENUM('Disponível', 'Emprestado', 'Danificado') DEFAULT 'Disponível',  
  PRIMARY KEY (codigoISBN, numeroDeCopia),  
  FOREIGN KEY (codigoISBN) REFERENCES Livro(codigoISBN)  
);
```

```
INSERT INTO Utilizador (nome, tipo, contacto)  
VALUES ('Joana Silva', 'Aluno', 'joana.silva@mail.com');
```

```
INSERT INTO Funcionario (nome, funcao)  
VALUES ('Carlos Mendes', 'Bibliotecário');
```

```
INSERT INTO Livro (codigoISBN, titulo, autor, anoDePublicacao, edicao, genero)  
VALUES ('9781234567897', 'A Máquina do Tempo', 'H.G. Wells', 1895, '1ª', 'Ficção Científica');
```

```
INSERT INTO Exemplar (codigoISBN, numeroDeCopia, estado)  
VALUES ('9781234567897', 1, 'Disponível');
```

```
INSERT INTO Localizacao (codigoISBN, numeroDeCopia, corredor, estante, prateleira)  
VALUES ('9781234567897', 1, 'B', '2', '3');
```

```
INSERT INTO Emprestimo (dataDeInicio, dataDeDevolucaoPrevista, numeroDeUtilizador, codigoInterno, codigoISBN, numeroDeCopia)  
VALUES ('2025-05-26', '2025-06-02', 1, 1, '9781234567897', 1);
```

MYSQL COM LOOPBACK

```
import {inject, lifecycleObserver, LifecycleObserver} from '@loopback/core';
import {juggler} from '@loopback/repository';

// const config = {
//   name: 'db',
//   connector: 'memory',
//   localStorage: '',
//   file: './data/db.json'
// };

// mysql
const config = {
  name: 'db',
  connector: 'mysql',
  url: '',
  host: 'localhost',
  port: 3306,
  user: 'root',
  password: 'ms@040576',
  database: 'biblioteca'
};

// Observe application's life cycle to disconnect the datasource when
// application is stopped. This allows the application to be shut down
// gracefully. The `stop()` method is inherited from `juggler.DataSource`.
// Learn more at https://loopback.io/doc/en/lb4/Life-cycle.html
@lifecycleObserver('datasource')
export class DbDataSource extends juggler.DataSource
  implements LifecycleObserver {
  static dataSourceName = 'db';
  static readonly defaultConfig = config;

  constructor(
    @inject('datasources.config.db', {optional: true})
    dsConfig: object = config,
  ) {
    super(dsConfig);
  }
}
```

CONFIGURAÇÃO

Configuramos uma conexão com um banco de dados MySQL no framework Loopback, na classe `db.datasource.ts`. A conexão é configurada localmente, com os parâmetros necessários. Também foi acrescentada uma `@lifecycleObserver` para que o loopback seja fechado a conexão automaticamente quando a aplicação for desligada.



PROBLEMAS ENCONTRADOS

LOOPBACK BELONGSTO

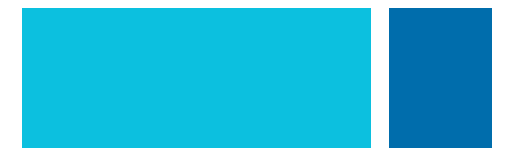
Embora os modelos tenham sido criados corretamente a partir das tabelas existentes, os relacionamentos entre as entidades — como chaves estrangeiras — não foram refletidos como associações explícitas nos modelos. O loopback não nos gerou um relation @belongsTo().

ENTIDADE EXEMPLARES

Na tabela exemplar, temos a chave primária composta por codigoISBN e numeroDeCopia. No entanto, o campo numeroDeCopia deve se repetir em toda a tabela — ou seja, pode haver exemplares com o mesmo numeroDeCopia, mesmo que tenham codigoISBN diferentes.

MYSQL HOST

O mySQL está sendo hosteado localmente, sendo difícil acesso em outra máquina.





SOLUÇÕES ENCONTRADAS

LOOPBACK BELONGSTO

Criar a adição manual do `@belongsTo()` nos modelos e repositórios..

ENTIDADE EXEMPLARES

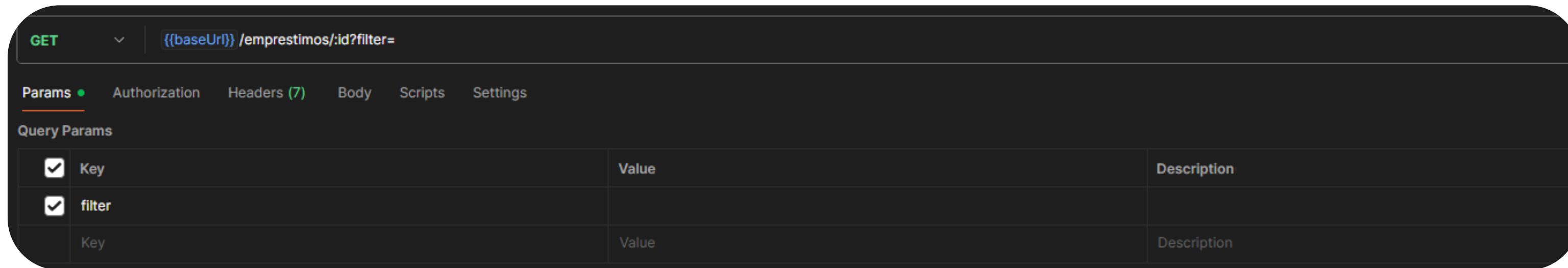
Implementação de `codigoISBN` como chave primária também.

MYSQL HOST

Infelizmente, não conseguimos resolver este problema.



POSTMAN - ORGANIZAÇÃO DOS ENDPOINTS



**Inclui todos os métodos
CRUD: GET, POST,
PATCH, PUT, DELETE.**

A coleção foi organizada por entidades, com os endpoints necessários para se gerir a biblioteca. Mesmo que haja muitas rotas.

EMPRESTIMO

Endpoint
GET /emprestimo
GET /emprestimo/ : id
GET /emprestimo/ : id/penalizacao ou com filter
POST , PUT , DELETE

EXEMPLAR

Endpoint
GET /exemplar
GET /exemplar/ : id
GET /exemplar/ : id/localizacao ou com filter
POST , PUT , DELETE

POSTMAN

ORGANIZAÇÃO GERAL

FUNCIONARIO

Endpoint
GET /funcionario
GET /funcionario/ : id
GET /funcionario/ : id/emprestimo
GET /funcionario/ : id/reserva
POST , PUT , DELETE

LIVRO

Endpoint
GET /livro
GET /livro/ : id
GET /livro/ count
GET /livro/ : id/emprestimo
GET /livro/ : id/exemplar
GET /livro/ : id/reserva
POST , PUT , DELETE

LOCALIZACAO

Endpoint
GET /localizacao
GET /localizacao/ : id
GET /localizacao/ count
POST , PUT , DELETE

PENALIZACAO

Endpoint
GET /penalizacao
GET /penalizacao/ : id
GET /penalizacao/ count
POST , PUT , DELETE

POSTMAN

ORGANIZAÇÃO GERAL

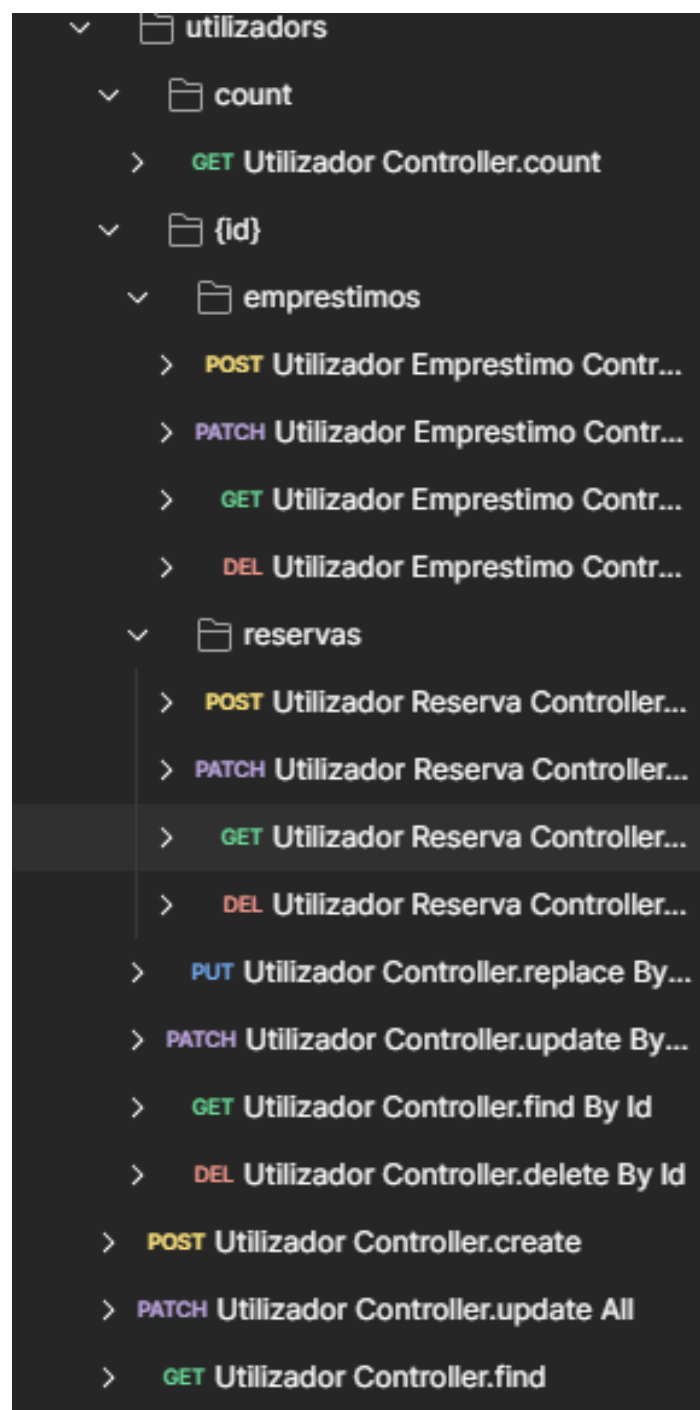
RESERVA

Endpoint
GET /reserva
GET /reserva/ : id
GET /reserva/ count
POST , PUT , DELETE

UTILIZADOR

Endpoint
GET /utilizador
GET /utilizador/ : id
GET /utilizador/ count
GET /utilizador/ : id / emprestimo
GET /utilizador/ : id / reserva
POST , PUT , DELETE

ORGANIZAÇÃO ENDPOINTS E CÓDIGO



```
1 import {
2   Count,
3   CountSchema,
4   Filter,
5   FilterExcludingWhere,
6   repository,
7   Where,
8 } from '@loopback/repository';
9 import {
10   post,
11   param,
12   get,
13   getModelSchemaRef,
14   patch,
15   put,
16   del,
17   requestBody,
18   response,
19 } from '@loopback/rest';
20 import {Utilizador} from '../models';
21 import {UtilizadorRepository} from '../repositories';
22
23 export class UtilizadorController {
24   constructor(
25     @repository(UtilizadorRepository)
26     public utilizadorRepository : UtilizadorRepository,
27   ) {}
28
29   @post('/utilizadores')
30   @response(200, {
31     description: 'Utilizador model instance',
32     content: {'application/json': {schema: getModelSchemaRef(Utilizador)}},
33   })
34   async create(
35     @requestBody({
36       content: {
37         'application/json': {
38           schema: getModelSchemaRef(Utilizador, {
39             title: 'NewUtilizador',
40             exclude: ['numeroDeUtilizador'],
41           }},
42       },
43     })
44     utilizador: Omit<Utilizador, 'numeroDeUtilizador'>,
45   ): Promise<Utilizador> {
46     return this.utilizadorRepository.create(utilizador);
47   }
48
49   @get('/utilizadores/count')
50   @response(200, {
51     description: 'Utilizador model count',
52     content: {'application/json': {schema: CountSchema}},
53   })
54   async count(
55     @param.where(Utilizador) where?: Where<Utilizador>,
56   ): Promise<Count> {
57     return this.utilizadorRepository.count(where);
58   }
59 }
```

<div> <div> <div></div> <div>utilizadores</div> </div> <div> <div></div> <div>count</div> </div> <div> <div></div> <div>Utilizador Controller.count</div> </div> </div>	
<div> <div> <div></div> <div>{Id}</div> </div> <div> <div></div> <div>emprestimos</div> </div> <div> <div></div> <div>Utilizador Emprestimo Contr...</div> </div> <div> <div></div> <div>Utilizador Emprestimo Contr...</div> </div> <div> <div></div> <div>Utilizador Emprestimo Contr...</div> </div> <div> <div></div> <div>Utilizador Emprestimo Contr...</div> </div> </div>	
<div> <div> <div></div> <div>reservas</div> </div> <div> <div></div> <div>Utilizador Reserva Controller...</div> </div> <div> <div></div> <div>Utilizador Reserva Controller...</div> </div> <div> <div></div> <div>Utilizador Reserva Controller...</div> </div> <div> <div></div> <div>Utilizador Reserva Controller...</div> </div> <div> <div></div> <div>Utilizador Controller.replace By...</div> </div> <div> <div></div> <div>Utilizador Controller.update By...</div> </div> <div> <div></div> <div>Utilizador Controller.find By Id</div> </div> <div> <div></div> <div>Utilizador Controller.delete By Id</div> </div> <div> <div></div> <div>Utilizador Controller.create</div> </div> <div> <div></div> <div>Utilizador Controller.update All</div> </div> <div> <div></div> <div>Utilizador Controller.find</div> </div> </div>	

```

1 import {
2   Count,
3   CountSchema,
4   Filter,
5   FilterExcludingWhere,
6   repository,
7   Where,
8 } from '@loopback/repository';
9 import {
10   post,
11   param,
12   get,
13   getModelSchemaRef,
14   patch,
15   put,
16   del,
17   requestBody,
18   response,
19 } from '@loopback/rest';
20 import {Utilizador} from '../models';
21 import {UtilizadorRepository} from '../repositories';
22
23 export class UtilizadorController {
24   constructor(
25     @repository(UtilizadorRepository)
26     public utilizadorRepository : UtilizadorRepository,
27   ) {}
28
29   @post('/utilizadores')
30   @response(200, {
31     description: 'Utilizador model instance',
32     content: {'application/json': {schema: getModelSchemaRef(Utilizador)}},
33   })
34   async create(
35     @requestBody({
36       content: {
37         'application/json': {
38           schema: getModelSchemaRef(Utilizador, {
39             title: 'NewUtilizador',
40             exclude: ['numeroDeUtilizador'],
41           }},
42         },
43       },
44     ))
45     utilizador: Omit<Utilizador, 'numeroDeUtilizador'>,
46   ): Promise<Utilizador> {
47     return this.utilizadorRepository.create(utilizador);
48   }
49
50   @get('/utilizadores/count')
51   @response(200, {
52     description: 'Utilizador model count',
53     content: {'application/json': {schema: CountSchema}},
54   })
55   async count(
56     @param.where(Utilizador) where?: Where<Utilizador>,
57   ): Promise<Count> {
58     return this.utilizadorRepository.count(where);
59   }
60 }

```

POSTMAN

