Proyecto de Gestión de Contratos

-

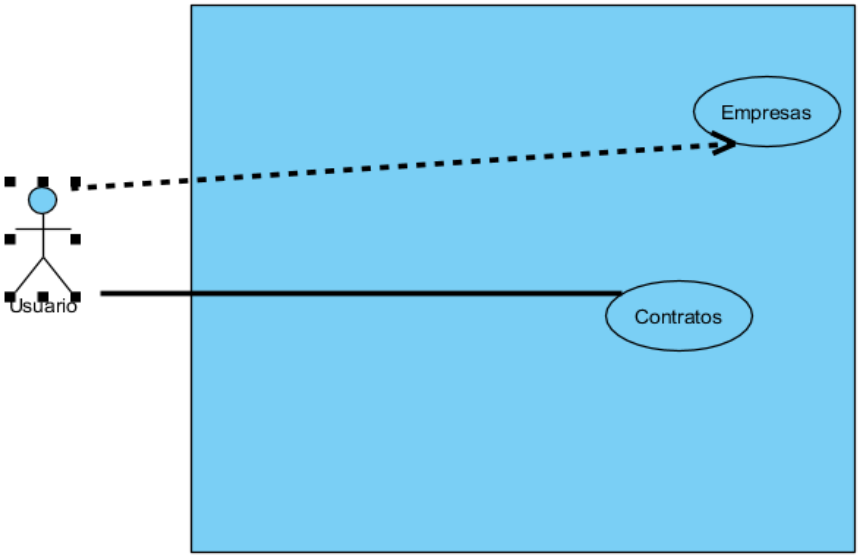
Subcontrata sector servicios.

Índice

Descripción del proyecto:

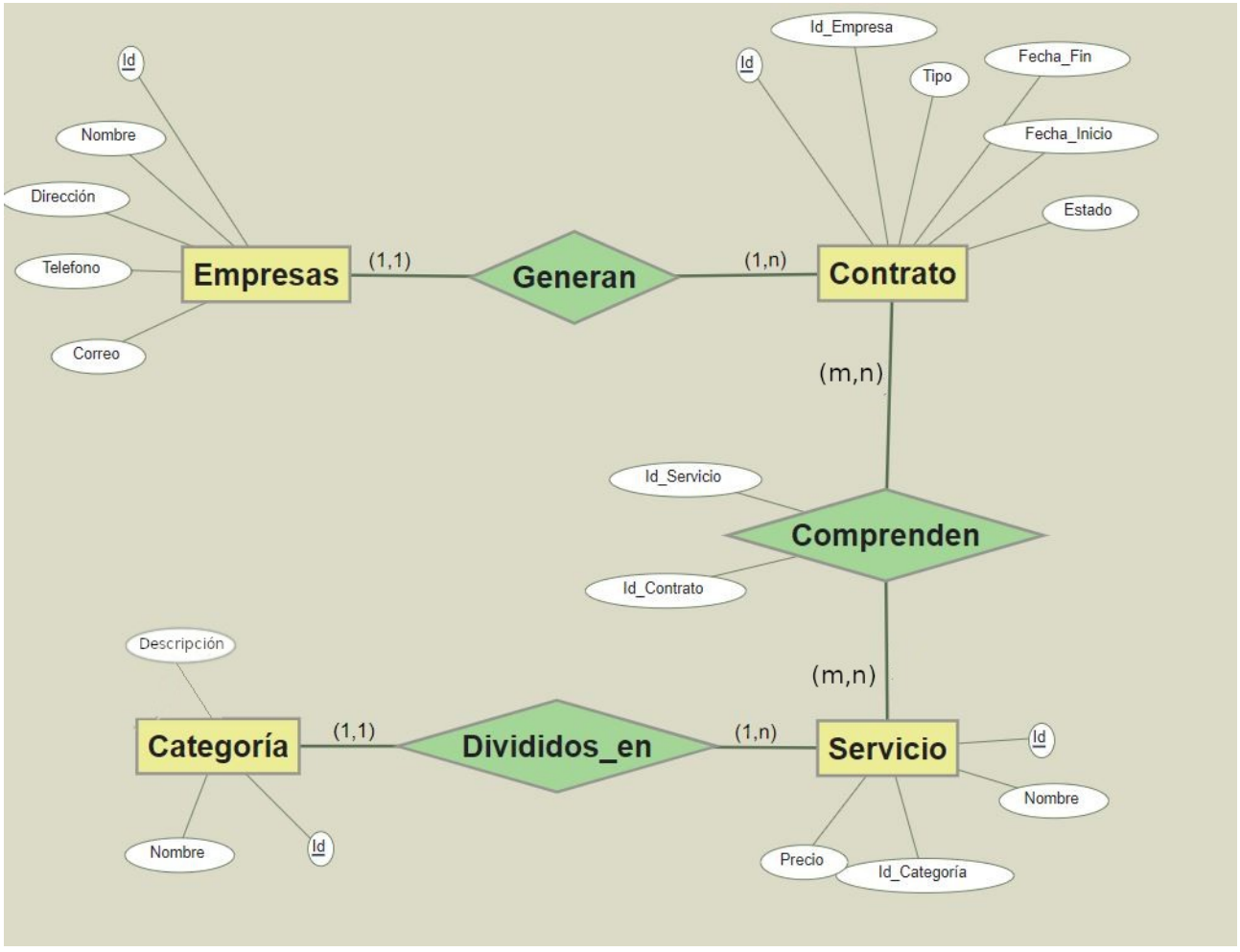
La meta de este proyecto es realizar una aplicación de gestión de contratos que cree, actualice, lea y borre instancias de los mismos para una cantidad finita de empresas y los cuales contendrán de uno a varios servicios. Estos los dividiremos en categorías.

Casos de Uso:



Base de datos MySQL:

Modelo E-R:



SQL e Inserts:

DROP DATABASE IF EXISTS subcontrata;

CREATE DATABASE subcontrata;

DROP TABLE IF EXISTS empresas;

DROP TABLE IF EXISTS contratos;

DROP TABLE IF EXISTS servicios;

DROP TABLE IF EXISTS comprenden;

CREATE TABLE empresas(

id INT AUTO_INCREMENT NOT NULL,

nombre VARCHAR(255) NOT NULL,

direccion VARCHAR(255) NOT NULL,

telefono VARCHAR(14) NOT NULL,

correo VARCHAR(100) NOT NULL,

CONSTRAINT pk_idEmpresa PRIMARY KEY (id)

);

CREATE TABLE contratos(

id INT AUTO_INCREMENT NOT NULL,

id_empresa INT NOT NULL,

tipo VARCHAR(255) NOT NULL,

fecha_inicio DATE NOT NULL,

fecha_fin DATE NOT NULL,

CONSTRAINT pk_idContrato PRIMARY KEY (id),

CONSTRAINT fk_idEmpresa FOREIGN KEY (id_empresa) REFERENCES empresas (id)

);

CREATE TABLE categorias(

id INT AUTO_INCREMENT NOT NULL,

nombre VARCHAR(255) NOT NULL,

descripción VARCHAR (255),

CONSTRAINT pk_idCategorias PRIMARY KEY (id)

);

CREATE TABLE servicios(

id INT AUTO_INCREMENT NOT NULL,

nombre VARCHAR(255) NOT NULL,

id_categoria INT NOT NULL,

precio DOUBLE NOT NULL,

CONSTRAINT pk_idServicio PRIMARY KEY (id),

CONSTRAINT fk_idCategoria FOREIGN KEY (id_categoria)

REFERENCES categorias (id)

);

CREATE TABLE comprenden(

id INT AUTO_INCREMENT NOT NULL,

id_contrato INT NOT NULL,

id_servicios INT NOT NULL,

CONSTRAINT fk_idContrato FOREIGN KEY (id_contrato)

REFERENCES contratos (id), CONSTRAINT pk_idComprenden PRIMARY KEY (id), CONSTRAINT fk_idServicios FOREIGN KEY (id_servicios) REFERENCES servicios (id)

);

Empresas:

INSERT INTO `empresas`(`nombre`, `direccion`, `telefono`, `correo`) VALUES ('EmpresaTest1','DirecciónTest1','TelefonoTest1','CorreoTest1');

INSERT INTO `empresas`(`nombre`, `direccion`, `telefono`, `correo`) VALUES ('EmpresaTest2','DirecciónTest2','TelefonoTest2','CorreoTest2');

INSERT INTO `empresas`(`nombre`, `direccion`, `telefono`, `correo`) VALUES ('EmpresaTest3','DirecciónTest3','TelefonoTest3','CorreoTest3');

INSERT INTO `empresas`(`nombre`, `direccion`, `telefono`, `correo`) VALUES ('EmpresaTest4','DirecciónTest4','TelefonoTest4','CorreoTest4');

Categorias:

INSERT INTO `categorias`(`nombre`, `descripcion`) VALUES ('CategoriaTest1','Test 1 de Categoría');

INSERT INTO `categorias`(`nombre`, `descripcion`) VALUES ('CategoriaTest2','Test 2 de Categoría');

INSERT INTO `categorias`(`nombre`, `descripcion`) VALUES ('CategoriaTest3','Test 3 de Categoría');

INSERT INTO `categorias`(`nombre`, `descripcion`) VALUES ('CategoriaTest4','Test 4 de Categoría');

Servicios:

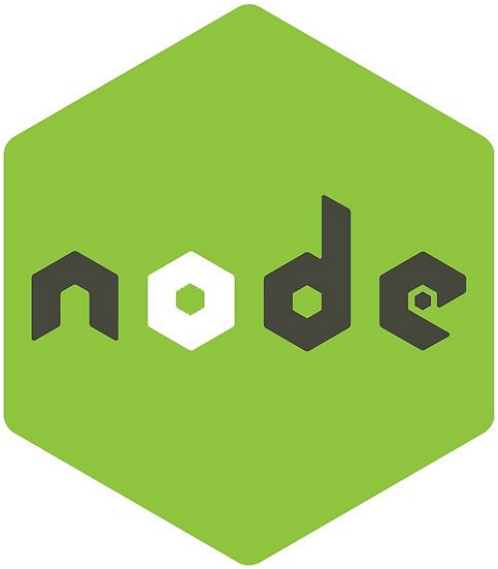
INSERT INTO `servicios`(`nombre`, `precio`, `categoriaId`) VALUES ('ServicioTest1',1.00,1);

INSERT INTO `servicios`(`nombre`, `precio`, `categoriaId`) VALUES ('ServicioTest2',2.00,2);

INSERT INTO `servicios`(`nombre`, `precio`, `categoriaId`) VALUES ('ServicioTest3',3.00,3);

INSERT INTO `servicios`(`nombre`, `precio`, `categoriaId`) VALUES ('ServicioTest4',4.00,4);

Servidor (Node js):



Por la parte de servidor contaremos con un CRUD en servidor para crear contratos y asignarle servicios y categorizarlos.

Para facilitar el trabajo en node ademas de instalar Sequelize como ORM añadiendo Nodemon, un paquete que me permite compilar cambios solo guardando el documento sobre el que trabajo sin tener que reiniciar, y Babel que permite codificar usando un lenguaje menos complicado y se traduce a código legible.

El servidor con sequelize nos permite una creación de las tablas usando modelos (Ej):

Modelos:

Empresas:

empresas.js

```
module.exports = (sequelize, DataType) => {
  const empresas = sequelize.define('empresas', {
    id: {
      type: DataType.INTEGER,
      primaryKey: true,
      autoIncrement: true
    },
    nombre: {
      type: DataType.STRING,
      allowNull: false,
      validate: {
        notEmpty: true
      }
    },
    direccion: {
      type: DataType.STRING,
      allowNull: false,
      validate: {
        notEmpty: true
      },
    },
    telefono: {
      type: DataType.STRING,
      allowNull: false,
      validate: {
        notEmpty: true
      },
    },
    correo: {
      type: DataType.STRING,
      allowNull: false,
      validate: {
        notEmpty: true
      }
    }
  }, {
    timestamps: false,
    freezeTableName: true
  });
  empresas.associate = (models) => {
    empresas.hasMany(models.contratos);
  };
  return empresas;
};
```

Contratos:

contratos.js

```
module.exports = (sequelize, DataType) => {
  const contratos = sequelize.define("contratos", {
    id: {
      type: DataType.INTEGER,
      primaryKey: true,
      autoIncrement: true
    },
    tipo: {
      type: DataType.STRING,
      allowNull: false,
      validate: {
        notEmpty: true
      }
    },
    fecha_inicio: {
      type: DataType.DATEONLY,
      allowNull: false
    },
    fecha_fin: {
      type: DataType.DATEONLY,
      allowNull: false
    }
  }, {
    timestamps: false,
    freezeTableName: true
  });
  contratos.associate = (models) => {
    contratos.belongsTo(models.empresas, {
      foreignkey: 'id_empresa'
    });
    contratos.hasMany(models.comprenden);
  };
  return contratos;
};
```

Categorías:

categorias.js

```
module.exports = (sequelize, DataType) => {
  const categorias = sequelize.define('categorias', {
    id: {
      type: DataType.INTEGER,
      primaryKey: true,
      autoIncrement: true
    },
    nombre: {
      type: DataType.STRING,
      allowNull: false,
      validate: {
        notEmpty: true
      }
    },
    descripcion: {
      type: DataType.STRING,
      allowNull: true
    },
  }, {
    timestamps: false,
    freezeTableName: true
  });
  categorias.associate = (models) => {
    categorias.hasMany(models.servicios);
  };
  return categorias;
};
```

Servicios:

servicios.js

```
module.exports = (sequelize, DataType) => {
  const servicios = sequelize.define("servicios", {
    id: {
      type: DataType.INTEGER,
      primaryKey: true,
      autoIncrement: true
    },
    nombre: {
      type: DataType.STRING,
      allowNull: false,
      validate: {
        notEmpty: true
      }
    },
    precio: {
      type: DataType.DOUBLE,
      allowNull: false
    }
  }, {
    timestamps: false,
    freezeTableName: true
  });
  servicios.associate = (models) => {
    servicios.belongsTo(models.categorias);
    servicios.hasMany(models.comprenden);
  };
  return servicios;
};
```

Comprenden:

comprenden.js

```
module.exports = (sequelize, DataType) => {
  const comprenden = sequelize.define("comprenden", {
    id: {
      type: DataType.INTEGER,
      primaryKey: true,
      autoIncrement: true
    }
  }, {
    timestamps: false,
    freezeTableName: true
  });
  comprenden.associate = (models) => {
    comprenden.belongsTo(models.contratos);
    comprenden.belongsTo(models.servicios);
  };
  return comprenden;
};
```

Y gestiona las operaciones a realizar(La creación, lectura, edición y borrado) sobre las mismas usando rutas (Ej):

Rutas-(Postman):

https://documenter.getpostman.com/view/6052118/Rzfnkn2Z

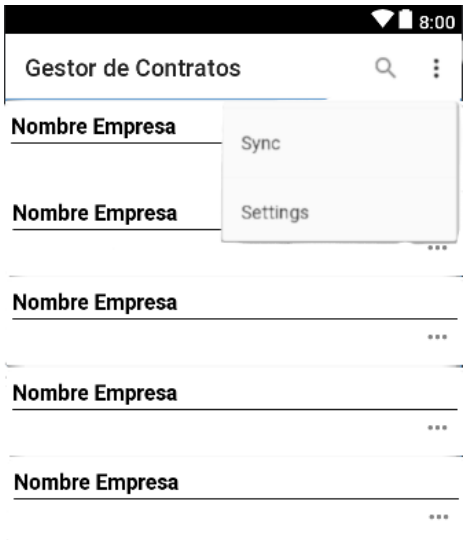
Todo mediante el trafico de objetos JSON.

Aplicación del Cliente(Android Studio):

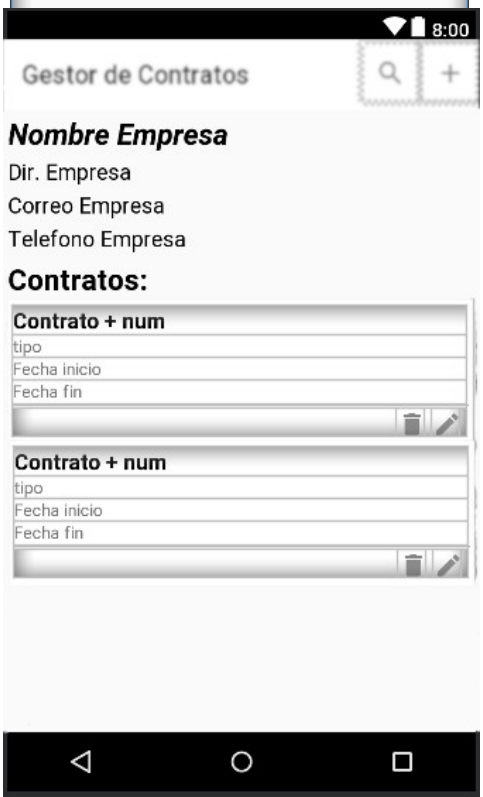
Por último, el cliente móvil estará desarrollado en Android Studio usando el lenguaje de programación Kotlin.

Contará de 3 pantallas*:

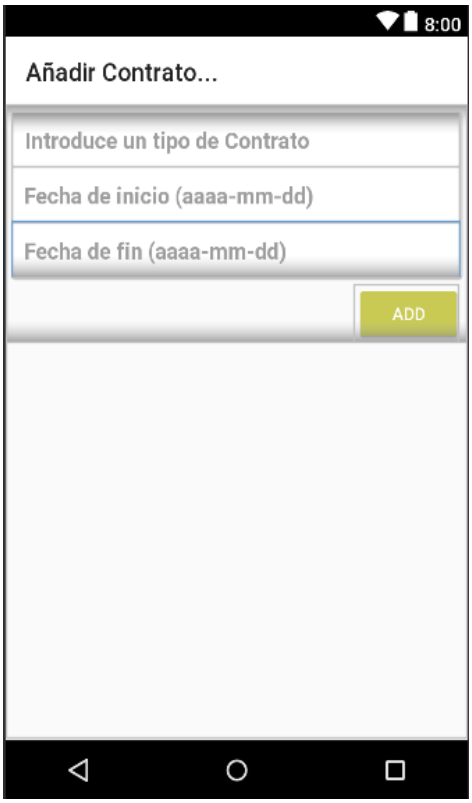
- Una con todas las empresas en la base de datos



- La segunda con los detalles de la empresa seleccionada y sus contratos.



- La tercera es el formulario con la creación de los contratos para la empresa anteriormente elegida



A parte implementa una base de datos local, en la pantalla inicial existe un botón para sincronizar con la base de datos usando el ORM (Pendiente de hacerlo automáticamente) y una barra de búsqueda