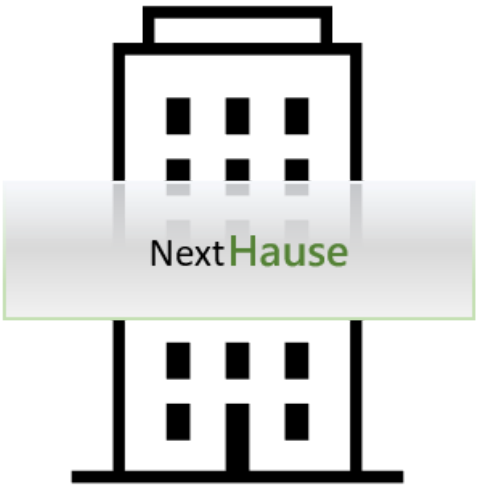
NextHause



Relatório Final - Base de Dados

Dário Matos (89288) e Pedro Almeida (89205)

universidade de aveiro

1. Análise de Requisitos

A ideia do projeto provém da complexidade conhecida em gerir um condomínio/contratos de renda de imóveis, tendo como principal objetivo facilitar a manipulação de informação deste contexto.

O NextHause visa representar um sistema capaz de suportar informação acerca de:

- **Pessoas** (sob a forma de Inquilinos, Fiadores, ou Proprietários) identificados pelo Número de Identificação Fiscal, contendo características como o nome, telefone, Número do Cartão de Cidadão e, em casos de específicos, Registo Criminal, Certificado de Salário, etc..

- **Casas**, caracterizadas por número máximo de habitantes, número de quartos, cidade e condomínio, identificadas exclusivamente pela sua morada.

- **Condomínios**, associados a múltiplas Casas, identificados também pelo NIF.

- **Contratos** (de Condomínio ou de Renda). No primeiro caso, o proprietário representa quem habita no imóvel associado ao condomínio, e no segundo, quem possui o imóvel. Caracterizados por data de início e fim, podendo ter Despesa, Seguro e Área do Imóvel (Condomínio) ou Valor de Renda, Caução, Taxa, Fiador, Inquilino e Empresa (Renda), todos os contratos são identificados por um código.

- **Obras** nos condomínios, definidas num intervalo de tempo e com um determinado orçamento, identificadas por um ID.

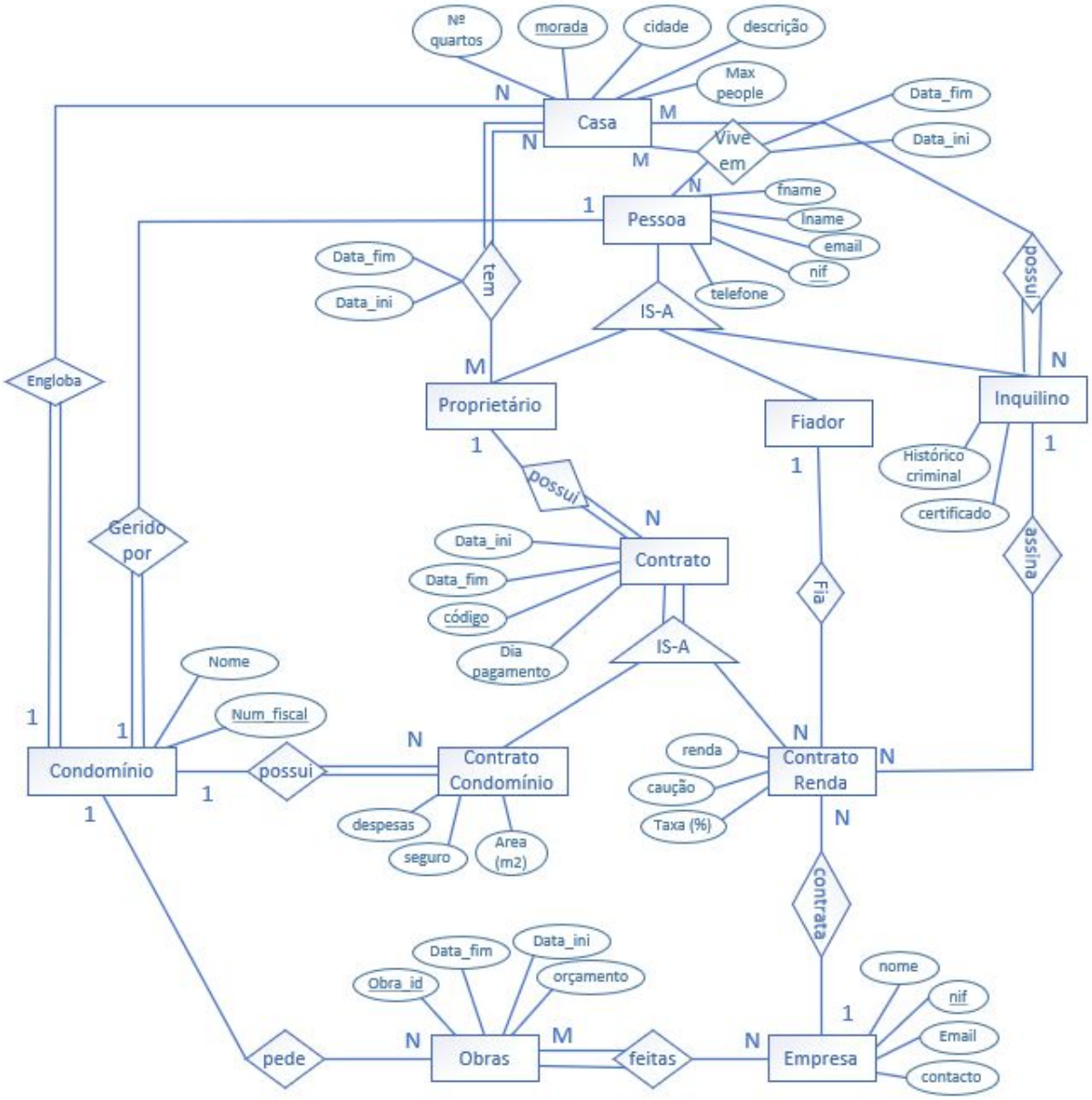
- **Empresas** responsáveis pelo arrendamento de imóveis ou por Obras feitas em condomínios, caracterizadas por nome, contacto telefónico e email, identificadas também pelo seu NIF.

No caso dos Proprietários, o sistema admite a possibilidade de obter mais do que uma Casa, e vice versa, relacionando estas entidades numa tabela específica(*tem_casa*). O mesmo acontece com os Inquilinos (*casa_inquilino*) e

uma Pessoa (*vive_em*), no caso de habitar e não possuir imóveis, nem pagar renda.

O NextHause tem também como principal funcionalidade a ligação de Condomínios com as Obras neles realizados e Empresas envolvidas (*faz_obras*) e guardar um registo de contratos entre pessoas e proprietários.

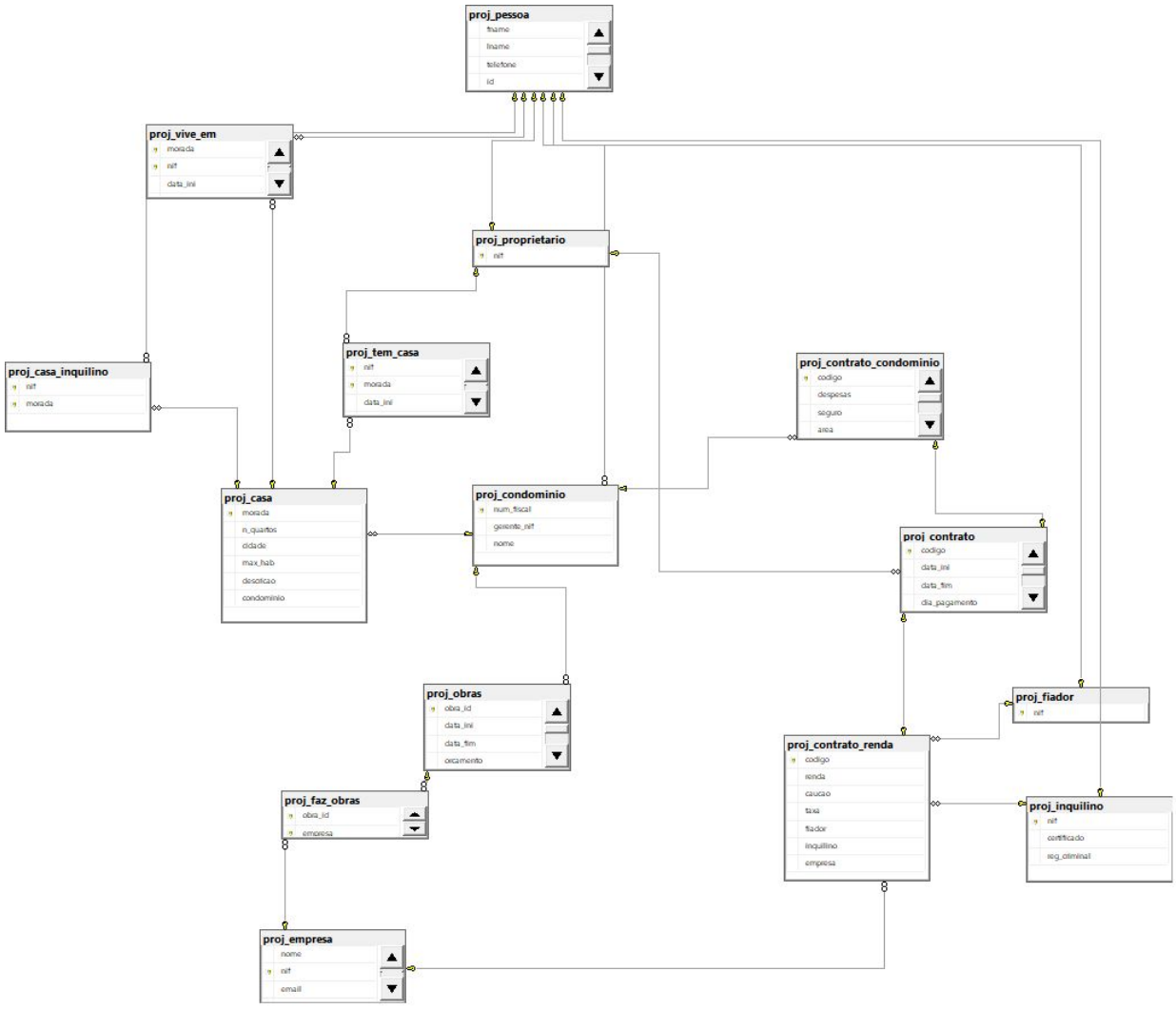
2. DER



No desenvolvimento da segunda fase do projeto foram feitas pequenas alterações que são visíveis no Desenho Conceptual, sendo essas alterações:

- Adição do atributo *Nome* na entidade *Condomínio*. Surgiu na necessidade de o utilizador conseguir identificar o Condomínio sem ser pelo seu *Número Fiscal*.
- Remoção do atributo *n_renda* (número de renda) na entidade *Contrato de Renda*. Seria mais um campo a preencher pelo o utilizador e não foi encontrada a utilidade do atributo.

3. Esquema Relacional



4. SQL DDL e DML

(em anexo)

5. Normalização

Não foram encontradas tabelas cuja normalização fosse necessária no decorrer do desenvolvimento deste projeto, encontrando-se todas na 3FN.

6. Views

De modo a apresentar os dados nas ListBox de uma forma mais interativa ao utilizador (mostrando nomes e não números) foram criadas as seguintes views:

- ❏ Moradas
- ❏ Show_Condominios
- ❏ Show_Empresas
- ❏ Show_Gerentes

7. Programação

- **Stored Procedures**

  Foram usados SPs para a inserção de dados, remoção de dados e obter informação com base em argumentos passados, garantindo desta maneira um processo mais eficiente e não dar azo a falhas de segurança como SQL Injection.

  No caso de atualização de dados de mais do que uma tabela, foram ainda usadas **Transactions**. Deste modo é garantido que caso haja algum erro na inserção de dados de umas das tabelas, é possível retorcer a uma situação segura.

Os SPs criados foram:

- deleteCasa
- deletePerson
- getCasaByMorada
- getCasaByMorada2
- getCasasByCondominio
- getCondominio
- getEmpresaByNif
- getFnameLname
- getMoradasCasasByCondominio
- GetNifByFnameLname
- getNomeCond
- getNomeEmpresaByNif
- getNumFiscalCond
- getObrasById
- getPessoasByCasa
- getPropsByCasa
- inserirCasa
- inserirContratoCond
- inserirContratoRenda
- inserirFiador
- inserirInq
- insertCond
- insertProp
- insertCond
- insertEmpresa
- insertObra
- updateCasa
- updateCond
- updateEmpresa
- updateObra

Relativamente ao “deletePerson”, a remoção de uma Pessoa do sistema não compromete os contratos por ela assinados, alterando apenas o campo dos mesmos para “null” na base de dados.

- **UDFs (User Defined Functions)**
  Para além da melhoria em termos de performance à semelhança dos SPs, os propósitos para a utilização de UDFs

foram a possibilidade de estes últimos poderem ser usados como fonte de dados.

Foram usados dois tipos de UDFs:

- Table-value Functions
  - gastos_condominio
  - getInquilinosByCond
  - getObras
  - getPessoasByCond
  - getProprietariosByCond

- Scalar-value Functions:
  - gastoTotal

Na construção de alguns UDFs foram também usados **Cursores**.

- **Triggers**
  - check_dates - Verifica que a data final do contrato é depois da data inicial

  - check_datesObras - Verifica que a data final da obra é depois da data inicial

  - validInsertObra - verifica se os dados a inserir não são duplicados

8. Alterações perante a apresentação em Aula Prática

O trabalho realizado depois da apresentação foi:

- Adição da funcionalidade de eliminação de *Pessoas* e *Casas*;
- Adição da funcionalidade de adicionar *Condomínios*;
- Adição da funcionalidade de adicionar *Fiadores*;

- Adição da funcionalidade de adicionar e editar *Obras*;
- Passagem de todo o SQL DML para SPs;
- Alteração das mensagens de erro para melhorias a nível de segurança;
- Mudanças visuais da interface;
- Correção de diversos bugs encontrados ao longo do desenvolvimento do projeto;
- Alteração do vídeo de apresentação;

9. Nota

Para alterar as credenciais de acesso à base de dados deve alterar no ficheiro App.config o atributo *connectionString* (linha 4) para o desejado.

10. Conclusão

A realização deste projeto leva-nos a concluir o papel crucial que desempenha uma base de dados robusta e organizada no bom funcionamento de serviços da vida quotidiana, bem como a complexidade da mesma.
Acreditamos que todas as técnicas assimiladas na cadeira de Base de Dados levam a compreender melhor como gerir os dados de sistemas presentes nas mais diversas áreas.

Para lá dos requisitos inicialmente planeados, o grupo acredita que pode ainda ser feito algum trabalho para melhorar a interface do projeto NextHause, e serem adicionadas algumas funcionalidades como a marcação de rendas pagas ou em dívida e a consulta de preço a pagar/pago por pessoa. Posto isto, seria interessante introduzir uma versão melhorada do projeto no contexto profissional de gestão de condomínios e arrendamentos.