

Projeto de Bases de Dados (CC2005)

Grupo nº 62

Tema: Contratos Públicos em Portugal nos primeiros dois meses de 2024

Elementos do grupo:

Nº mecanográfico	Nome
202400891	Ezequiel Tchimbaya Cachapeu Paulo
202309160	Sérgio Gomes Pinto
202400863	Victor De Vargas Lopes

1. Modelação

1.1. Descrição do universo e modelo ER

1.1.1. Descrição do Universo da Base de Dados (Requisitos Textuais)

Requisitos do Universo do Discurso

Partimos por analisar a tabela única do aquivo dado no formato Excel e em paralelo o arquivo no formato txt (contendo a informação sumária da BD), onde derivamos o seguinte **conjunto de requisitos** que descrevem o universo de informações representado pela nossa BD:

- O sistema armazena informação sobre **Contratos Públicos em Portugal, com foco nos contratos que possuem data de celebração nos primeiros dois meses de 2024**, cada um identificado por um *idContrato* único.
- A base de dados contém contratos no intervalo entre dezembro de 2023 e fevereiro de 2024, abrangendo tanto o período de divulgação quanto de celebração dos contratos.
- Os contratos são predominantemente em território português, mas também existem contratos com locais de execução em outros países, como Espanha, Alemanha, Estados Unidos, Arábia Saudita, Cabo Verde, dentre outros. Por esse motivo, os locais de execução possuem informação detalhada de município, distrito e país.

- Cada contrato possui:
 - um **IdContrato**, contendo um número de identificação único para cada contrato.
 - um **tipo de contrato** (ex.: aquisição de bens móveis, aquisição de serviços, empreitadas de obras públicas e etc.).
 - um **tipo de procedimento**, indicando a forma de contratação.
 - uma **descrição do objeto do contrato**, correspondendo ao bem ou serviço adquirido.
 - uma **data de publicação** na base nacional de contratos.
 - uma **data de celebração**, que define quando o contrato foi formalmente celebrado.
 - um **preço contratual**.
 - um **CPV** (Código de Vocabulário Comum para Contratos Públicos).
 - um **prazo de execução**.
- Cada contrato envolve uma e **apenas uma entidade adjudicante**, a qual:
 - possui **número fiscal (NIF)**.
 - possui **designação oficial**.
- Um contrato pode envolver **um ou vários adjudicatários** (fornecedores), cada um dos quais:
 - possui **número fiscal (NIF)**.
 - possui **designação oficial**.
- O sistema armazena a **localização da execução do contrato**, devendo esta ser desagregada em:
 - país
 - distrito
 - município

Cada uma destas localizações deve ser representada como entidade própria com **códigos oficiais** (ex.: Pais_id, Distrito_id, Municipio_id).
O sistema permite múltiplas localizações de execução para um mesmo contrato.
- O sistema armazena a **fundamentação** da contratação: a justificação formal para a escolha do procedimento e/ou adjudicatário.
- O sistema indica se o contrato resulta de um **procedimento centralizado**.
- Se aplicável, o sistema regista o **Acordo Quadro** que enquadra o contrato, devendo armazenar a sua designação.

1.1.2. Concretização em Modelo ER (Modelo conceitual derivado da forma textual)

Alistando todas as colunas da tabela (dataset original), temos:

IdContrato, tipoContrato, tipoprocedimento, objectoContrato, adjudicante, adjudicatarios, dataPublicacao, dataCelebracaoContrato, precoContratual, cpv, prazoExecucao, localExecucao, fundamentacao, ProcedimentoCentralizado e DescrAcordoQuadro.

Durante as aulas, aprendemos que a presença de uma única tabela geralmente significa que **várias entidades estão “achatadas”** em uma só. Devemos, entretanto, examinar a tabela (única), buscando entender:

- Quais colunas existem
- Que tipo de informação cada coluna contém
- Quais grupos de colunas parecem pertencer a entidades diferentes
- Como as linhas se repetem e etc.

Nesta etapa, o nosso foco está voltado para reconhecer “pistas” sobre entidades e relacionamentos.

Para separar as devidas entidades, guiamo-nos por aspectos como:

- Colunas que descrevem pessoas
- Colunas que descrevem produtos
- Colunas que descrevem eventos (ex.: vendas, celebração, publicação, execução)
- Colunas que pertencem a lugares, datas, categorias, etc.

A estrutura de entidades-tipo decorrentes dos requisitos é dada por:

1.1.2.1. Entidades-tipo

A tabela original (dada) mistura **cinco entidades principais** (Contrato, Entidade, CPV, Localização e AcordoQuadro) com seus respectivos atributos:

1. CONTRATO

- **Atributos:** IdContrato, TipoContrato, TipoProcedimento, ObjectoContrato, DataPublicacao, DataCelebracaoContrato, PrecoContratual, PrazoExecucao, Fundamentacao, ProcedimentoCentralizado.

2. ENTIDADE

(Representa adjudicantes e adjudicatários)

- **Atributos:** Entidade_id, Nif, Designacao
- O atributo “Nif” é caracterizado como um valor **único**.
- Uma entidade pode desempenhar papéis diferentes em diferentes contratos.

3. CPV

- **Atributos:** Cpv_id, Codigo, Descrição

4. LOCALIZAÇÃO

(Seus **atributos** são multivalores compostos)

a) PAÍS

- Pais_Id, Nome

b) Distrito

- Distrito_id, Nome

c) MUNICÍPIO

- Municipio_id, Nome

5. ACORDOQUADRO

- **Atributos:** AcordoQuadro_id, DescrAcordoQuadro

1.1.2.2. Relacionamentos

Os respectivos relacionamentos decorrentes entre as Entidades_tipo são dadas por:

1. ADJUDICA (ENTIDADE (Adjudicante), CONTRATO)

- Um contrato tem **um e apenas um adjudicante**
- Uma entidade pode adjudicar/conceder **vários contratos**
- **Tipo:** 1:N

2. ADJUDICADO (CONTRATO, ENTIDADE (Adjudicatário))

- Um contrato pode ter **um ou vários adjudicatários**
- Uma entidade pode ser adjudicatária em **muitos contratos**
- **Tipo:** M:N via *AdjudicatarioContrato*

3. EXECUTADO_EM (CONTRATO, LOCALIZAÇÃO)

- Um contrato pode ter **uma ou várias localizações de execução**
- Uma localização (município) pode aparecer em vários contratos
- **Tipo:** M:N via *ContratoLocalExecucao*

4. POSSUI (CONTRATO, CPV)

- Cada contrato tem um ou mais códigos CPVs
- Um código CPV pode ser associado a muitos contratos
- **Tipo:** M:N

5. ENQUADRADO_POR (CONTRATO, ACORDOQUADRO)

- Um contrato pode ou não estar associado a um acordo quadro
- Um acordo quadro pode enquadrar muitos contratos
- **Tipo:** N:1

Por tabela:

BD Contratos: Cardinalidade (C) e Participação (P)		
Relacionamento	C	P
ADJUDICA (ENTIDADE (Adjudicante), CONTRATO)	1<>N	Parcial <> Total
ADJUDICADO (CONTRATO, ENTIDADE (Adjudicatário))	M<>N	Total <> Parcial
EXECUTADO_EM (CONTRATO, LOCALIZAÇÃO)	M<>N	Total <> Parcial
POSSUI (CONTRATO, CVP)	M<>N	Total <> Parcial
ENQUADRADO_POR (CONTRATO, ACORDOQUADRO)	N<>1	Parcial <> Total

1.1.3. Diagrama do Modelo ER

Apesar da existência de diversas ferramentas existentes para auxílio de ilustração de diagramas do modelo ER, nós(equipa) optamos pelo uso das ferramentas online [dbdia](#) e [draw.io](#) disponibilizadas pelo professor, por ser mais intuitivo (para a equipa).

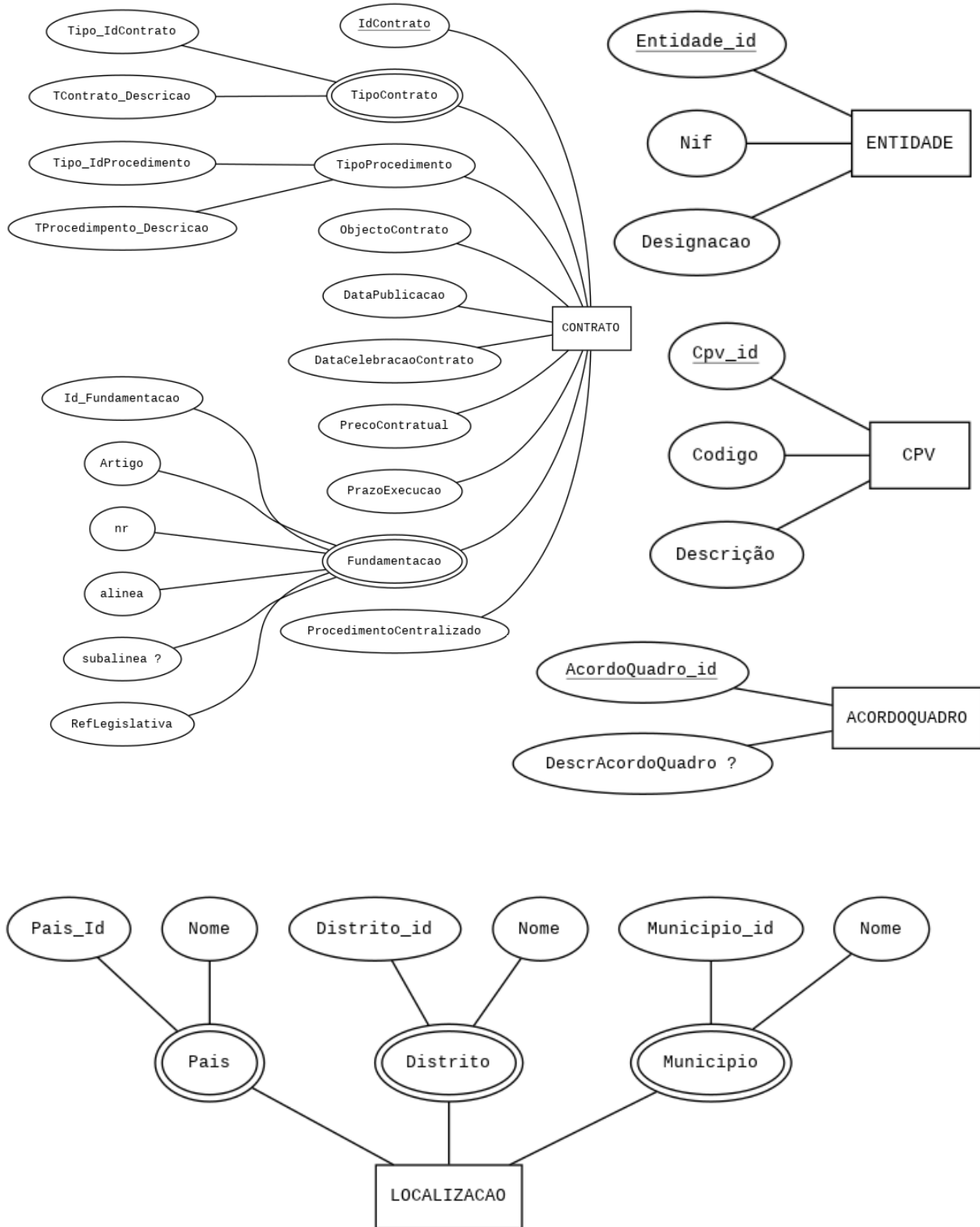
1.1.3.1. Para Entidades-tipo (com atributos principais)

Sintaxe Textual:

CONTRATO(IdContrato, {TipoContrato(Tipo_IdContrato, TContrato_Descricao)},
TipoProcedimento(Tipo_IdProcedimento, TProcedimento_Descricao), ObjectoContrato,
DataPublicacao, DataCelebracaoContrato, PrecoContratual, PrazoExecucao,
{Fundamentacao(Id_Fundamentacao, Artigo, nr, alinea, subalinea ?, RefLegislativa)},
ProcedimentoCentralizado)

ENTIDADE(Entidade_id, Nif, Designacao)
CPV(Cpv_id, Codigo, Descrição)
LOCALIZACAO({Pais(Pais_Id, Nome)}, {Distrito(Distrito_id, Nome)},
{Municipio(Municipio_id, Nome)})
ACORDOQUADRO(AcordoQuadro_id, DescrAcordoQuadro ?)

Diagrama:



1.1.3.2. Para os Relacionamentos (omitindo atributos internos para facilitar a visualização estrutural)

Sintaxe Textual:

```
ENTIDADE -- 1 -- <ADJUDICA> == N == CONTRATO
CONTRATO == M == <ADJUDICADO> -- N -- ENTIDADE
CONTRATO == M == <EXECUTADO_EM> -- N -- LOCALIZACAO
CONTRATO == M == <POSSUI> -- N -- CPV
CONTRATO -- N -- <ENQUADRADO_POR> == 1 == ACORDOQUADRO
```

Diagrama:

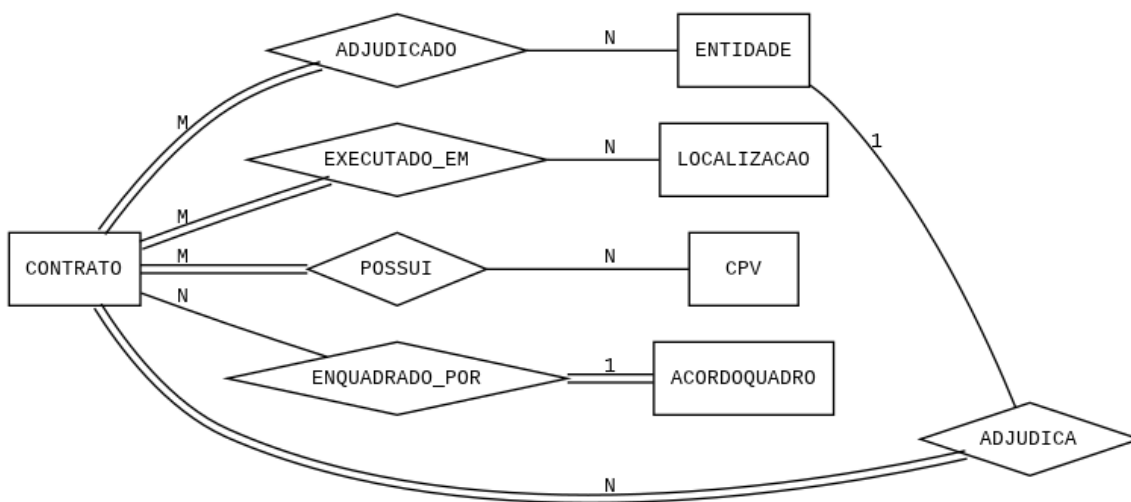


Diagrama Completo para o modelo ER:



1.2. Modelo relacional

1.2.1. Explicação do Mapeamento do Modelo ER para o Modelo Relacional

Em perspectiva geral o **Mapeamento** do Modelo ER para o Modelo Relacional consiste em transformar as **Entidades-tipo** e **Relacionamentos** (do modelo ER) em **tabelas** (para o modelo Relacional).

Para realizar tais transformações optamos realiza-las em etapas:

1.2.1.1. Mapeamento de entidades-tipo em tabelas

Nesta etapa, passamos pelo mapeamento directo de atributos, excepto no caso de atributos multi-valor em que precisamos de recorrer a “tabelas auxiliares” (no caso de *TipoContrato* e *Fundamentacao*), identificando assim no processo:

- Uma chave primária (**PK**)
- Atributos simples
- chaves externas (**FK**) apenas quando derivam de relacionamentos obrigatórios

Neste caso tornam-se tabelas:

Sintaxe Textual:

```
table CONTRATO
(
    _IdContrato_,
    Tipo_IdProcedimento,
    TProcedimento_Descricao,
    ObjectoContrato,
    DataPublicacao,
    DataCelebracaoContrato,
    PrecoContratual,
    PrazoExecucao,
    ProcedimentoCentralizado
)
```

```
table PAIS
(
    _Pais_Id_,
    Nome
)
```

```
table MUNICIPIO
(
    _Municipio_id_,
    Nome
)
```

```
table TipoContrato
(
    _Tipo_IdContrato_,
    TContrato_Descricao
)
```

```
table ENTIDADE
(
    _Entidade_nif_,
    Designacao
)
```

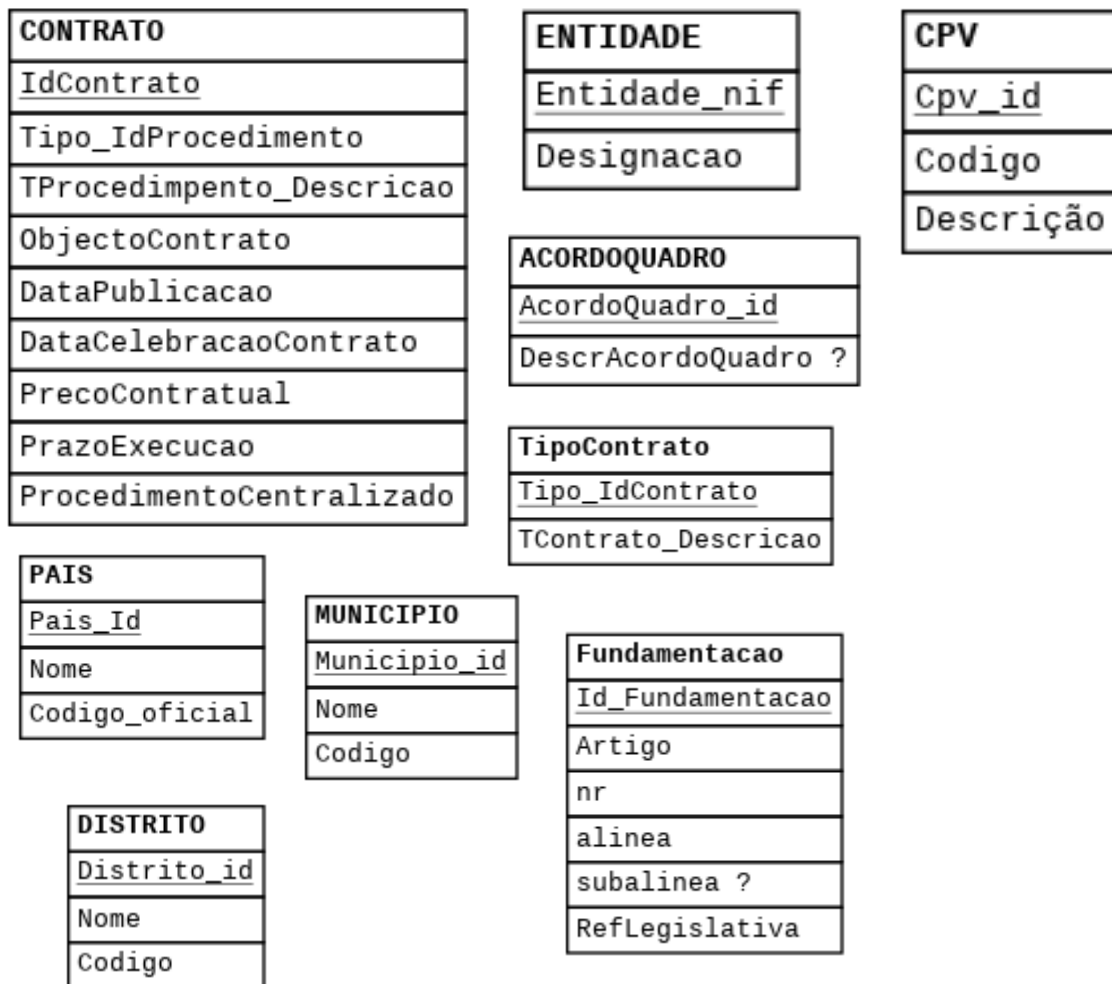
```
table CPV
(
    _Cpv_id_,
    Codigo,
    Descrição
)
```

```
table ACORDOQUADRO
(
    _AcordoQuadro_id_,
    DescrAcordoQuadro?
)
```

```
table DISTRITO
(
    _Distrito_id_,
    Nome,
)
```

```
table Fundamentacao
(
    _Id_Fundamentacao_,
    Artigo,
    nr,
    alinea,
    subalinea ?,
    RefLegislativa
)
```

Ilustração:



1.2.1.2. Mapeamento de relacionamentos em tabelas

Nesta etapa, passamos pelo mapeamento conforme a **cardinalidade** e **participação** das entidades-tipos de um relacionamento. Separamos em função de alguns critérios (para melhor entendimento):

1. Relacionamentos com cardinalidade 1:N

Em relações com essa cardinalidade em seus relacionamentos (no modelo ER), realizamos o respectivo mapeamento mantendo os atributos da Entidade-tipo do lado (N) adicionado do atributo-chave da Entidade-tipo do lado (1) como chave externa **FK** na **tabela** do lado N.

Sendo:

- *Entidade (1) – (N) Contrato (adjudicante) → Contrato* recebe adjudicante_id
- *(N) Contrato – AcordoQuadrado (1) → Contrato* recebe acordoquadrado_id
- *País (1) – (N) Distrito → Distrito* recebe pais_id como **FK**
- *Distrito (1) – (N) Município → Município* recebe distrito_id como **FK**

2. Relacionamentos com cardinalidade M:N

Em relações com essa cardinalidade em seus relacionamentos (no modelo ER), realizamos o respectivo mapeamento para uma espécie de “**Tabelas Associativas**”. Dois relacionamentos produzem tabelas intermédias:

a) Contrato — Entidade (Adjudicatário)

→ cria tabela **ADJUDICATARIO_CONTRATO** sendo as Chaves-primária (**PK**) a junção das duas **PK** das duas Entidades (idContrato, entidade_id).

b) Contrato — Município (Local Execução)

→ cria tabela **CONTRATO_LOCAL_EXECUCAO** sendo de igual modo as Chaves-primária (**PK**) a junção das duas **PK** das duas Entidades (idContrato, municipio_id).

c) CPV — Contrato

→ cria tabela **Possui** sendo de igual modo as Chaves-primária (**PK**) a junção das duas **PK** das duas Entidades (CPV, Contrato).

3. Atributos não simples: com destaque aos que exigem um mapeamento por “Tabelas Associativas”:

1. Atributos compostos → desagregar

Partindo do mapeamento da Entidade-tipo **Localização** (já desagregado em 3 entidades) podemos obter o “localExecucao” das 3 entidades originárias:

- País
- Distrito
- Município

com relacionamentos hierárquicos 1:N.

Obs.: Por termos desagregado anteriormente os atributos **TipoContrato** e **Fundamentacao** (da entidade Contratos) dando origem a duas novas tabelas (como visto na ilustração anterior), as mesmas darão agora origem a duas novas “**tabelas associativas**” frutos da relação com as outras:

a) ClassificacaoContratos(TipoContrato, Contratos)

- **Tipo M:N**
- Um contrato pode ser clasificado por **um ou vários tipos de Contratos**.
- Um tipo de contrato pode classificar um ou vários contratos.

b) FundamentacaoContratos(Fundamentacao, Contratos)

- **Tipo M:N**
- Um contrato pode ser fundamentado por **uma ou mais fundamentações**.
- Uma fundamentação pode fundamentar um ou vários contratos.

2. Atributos multivalorados → nova tabela

O atributo multivalor do *localExecução* gera a tabela associativa **CONTRATO_LOCAL_EXECUCAO** sendo as Chaves-primária (**PK**) a junção das duas **PK** das duas Entidades (*idContrato*, *municipio_id*), com Chaves-externa (**FK**) de: **CONTRATO** (*idContrato*), **MUNICIPIO**(*municipio_id*).

1.2.1.3. MODELO RELACIONAL FINAL (Tabelas com PK e FK explícitas)

Neste caso, tornam-se tabelas resultantes do mapeamento:

Sintaxe Textual:

```
table ENTIDADE
(
    Entidade_id    PK,
    Nif,
    Designacao
)

table PAIS
(
    Pais_id        PK,
    Nome
)

table MUNICIPIO
(
    Municipio_id   PK,
    Nome,
    Distrito_id    FK → DISTRITO(distrito_id)
)

table CPV
(
    Cpv_id         PK,
    Codigo,
    Descricao
)

table DISTRITO
(
    Distrito_id    PK,
    Nome,
    Pais_id        FK → PAIS(Pais_id)
)

Table ACORDO_QUADRO
(
    AcordoQuadro_id PK,
    DescrAcordoQuadro
)

table POSSUI
(
    idContrato      PK, FK → CONTRATO(idContrato),
    Cpv_id          PK, FK → CPV(Cpv_id)
)

table ClassificacaoContratos
(
    idContrato      PK, FK → CONTRATO(idContrato),
    Tipo_IdContrato PK, FK → TipoContrato(Tipo_IdContrato)
)

table FundamentacaoContratos
(
    idContrato      PK, FK → CONTRATO(idContrato),
    Id_Fundamentacao PK, FK → Fundamentacao (Id_Fundamentacao)
)
```

```

table CONTRATO
(
    idContrato          PK,
    objectoContrato,
    dataPublicacao,
    dataCelebracaoContrato,
    precoContratual,
    prazoExecucao,
    procedimentoCentralizado,
    Tipo_IdProcedimento FK → CP TipoProcedimento(Tipo_IdProcedimento)
    adjudicante_id      FK → ENTIDADE(entidade_id),
    acordoQuadro_id      FK → ACORDO_QUADRO(acordoQuadro_id)
)

```

```

table TipoContrato
(
    Tipo_IdContrato PK,
    TContrato_Descricao
)

```

```

table ADJUDICATARIO_CONTRATO
(
    idContrato          PK, FK → CONTRATO(idContrato),
    entidade_id         PK, FK → ENTIDADE(entidade_id)
)

```

```

table CONTRATO_LOCAL_EXECUCAO
(
    idContrato          PK, FK → CONTRATO(idContrato),
    municipio_id        PK, FK → MUNICIPIO(municipio_id)
)

```

```

table Fundamentacao
(
    Id_Fundamentacao PK,
    Artigo,
    nr,
    alinea,
    subalinea ?,
    RefLegislativa
)

```

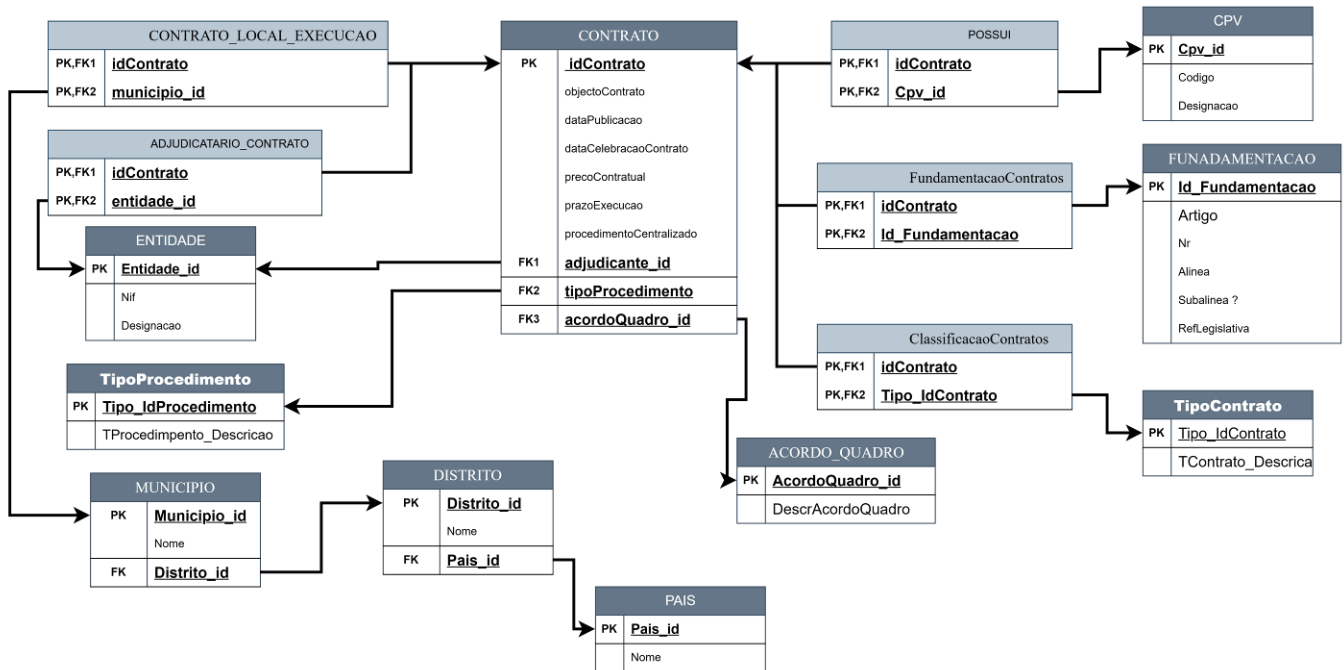
```

table TipoProcedimento
(
    Tipo_IdProcedimento PK,
    TProcedimento_Descricao
)

```

Diagrama:

Obs.: Para auxílio em ilustração (explicitando as respectivas chaves primárias e externas) de diagramas do Modelo Relacional, nós (equipa) optamos pelo uso da ferramenta online draw.io disponibilizada pelo professor, por ser mais intuitivo (para a equipa).



O respetivo modelo Relacional (final) fornece-nos segurança pois, para além de ser fruto do mapeamento directo do modelo ER, este está também apresentado na **3ª FN (Terceira Forma Normal)**, pois obedece as seguintes condições:

1º — Está em 1FN, isto é:

- Todos os atributos têm valores atômicos (sem atributos compostos nem multi-valores).

2º — Está em 2FN, isto é:

- Não existe dependência parcial (em nenhuma das tabelas com chave composta).

3º — Está em 3FN, isto é:

- **Não há dependências transitivas**

Ou seja:

um atributo não-chave não depende de outro atributo não-chave.

2. Implementação

2.1. Povoamento de tabelas

O povoamento da base de dados (contratos_publicos.db) foi realizado através do *script* Seed.py, seguindo rigorosamente a metodologia **ETL (Extract, Transform, Load)**. Este processo garantiu a migração de dados do *dataset* Excel para o modelo relacional normalizado.

O processo de Extract, Transform, Load foi implementado utilizando a biblioteca pandas para a fase de Extração, garantindo uma leitura rápida e eficiente do ficheiro Excel. A fase de Transformação foi a mais crítica, dado o formato não estruturado de vários campos da fonte de dados.

A criação das tabelas na base de dados foi realizada utilizando a linguagem SQL através de scripts Python, conforme definido no ficheiro scheme.py. Este processo envolveu a definição rigorosa da estrutura das tabelas, incluindo as chaves primárias (PK) e as chaves estrangeiras (FK), para garantir a integridade referencial dos dados (por exemplo, a ligação entre município, distrito e país).

Exemplo de criação de tabela simples:

```
-- PAIS
CREATE TABLE pais (
    pais_id INTEGER PRIMARY KEY AUTOINCREMENT,
    nome TEXT UNIQUE
);
```

Criação de uma tabela com uma chave estrangeira:

```
-- DISTRITO
CREATE TABLE distrito (
    distrito_id INTEGER PRIMARY KEY AUTOINCREMENT,
    nome TEXT,
    pais_id INTEGER,
    UNIQUE(nome, pais_id),
    FOREIGN KEY(pais_id) REFERENCES pais(pais_id)
);
```

A normalização da BD resultou na criação de tabelas intermediárias para modelar as relações (N:M). Estas tabelas contêm apenas duas chaves estrangeiras, sendo o par de chaves a sua Chave Primária Composta.

Normalização de Tipos: A conversão do campo precoContratual de texto para REAL implicou o tratamento explícito de separadores decimais (substituição de vírgula por ponto) e a gestão de valores nulos.

- **Parsing de Estruturas Múltiplas:** Campos como adjudicatários continham múltiplas entidades separadas por delimitadores (| ou ;). Foi desenvolvida uma função específica (split_adjudicatarios) que utiliza expressões regulares (re) para dividir o *string* em entidades individuais.
- **Tratamento de Entidades Complexas (Fundamentação):** A coluna fundamentacao exigiu a criação de uma função de *parsing* avançada (parse_fundamentacao) para isolar, via re, os constituintes (Artigo, Número, Alínea, etc.), garantindo que cada componente fosse armazenado na sua coluna dedicada na tabela fundamentação.

```
def parse_fundamentacao(texto):
    if pd.isna(texto) or texto is None:
        return None
    artigo = numero = alinea = subalinea = refLegislativa = None

    m = re.search(r"Artigo\s+([\dº\.]+)", texto, re.IGNORECASE)
    if m: artigo = m.group(1)
    m = re.search(r"n.º\s+(\d+)", texto, re.IGNORECASE)
    if m: numero = m.group(1)
    m = re.search(r"alínea\s+([a-z]\s+)", texto, re.IGNORECASE)
    if m: alinea = m.group(1)
    m = re.search(r"subalínea\s+([ivx]+\s+)", texto, re.IGNORECASE)
    if m: subalinea = m.group(1)
    m = re.search(r"do\s+(.*)$", texto, re.IGNORECASE)
    if m: refLegislativa = m.group(1).strip()

    return artigo, numero, alinea, subalinea, refLegislativa
```

- **Prevenção de Duplicidade (Controlo de Unicidade):** A função `insert_return_id` assegura que valores em tabelas de domínio (e.g., pais, cpv) são inseridos apenas uma vez. Se o registo já existir, o código recupera o ID existente (SELECT) em vez de tentar uma nova inserção, prevenindo erros de violação de UNIQUE e garantindo que todas as chaves estrangeiras (FK) apontam para um único registo.

```
def insert_return_id(sql_insert, params, sql_select, select_params):
    cur.execute(sql_select, select_params)
    row = cur.fetchone()
    if row: return row[0]
    cur.execute(sql_insert, params)
    return cur.lastrowid
```

1. Tratamento de Entidades (Adjudicante/Adjudicatários)

A tabela entidade foi especialmente adaptada para o tratamento de dados confidenciais (RGPD) e inconsistências no NIF:

- A **Chave Primária** passou a ser **entidade_id**, e o NIF tornou-se uma coluna de dados regular.
- A lógica de inserção foi ajustada para, na ausência de NIF, tentar associar o contrato à entidade existente usando apenas a designacao (nome), minimizando a criação de registos duplicados no caso de o NIF estar omissa.

Nome da tabela	Nº de entradas
Contrato	21748
Entidade	11698
Adjudicatario_Contrato	23282
CPV	2211
Possui	22067
Acordo_Quadro	256
Tipo_Contrato	8
ClassificacaoContratos	21911
Pais	18
Distrito	40
Municipio	348
Contrato_Local_Execucao	23317
Tipo_Procedimento	11
Fundamentacao	88
Fundamentacao_Contrato	21611

2.2. Interrogações SQL

Dividimos as 10 perguntas em 3 níveis de dificuldade, Fáceis, Médias e Difíceis

1. Fáceis

- Quais os tipos de contratos por ordem Alfabética?

```
SELECT nome
FROM tipo_contrato
ORDER BY nome
```

nome
Aquisição de bens móveis
Aquisição de serviços
Concessão de obras públicas
Concessão de serviços públicos
Empreitadas de obras públicas
Locação de bens móveis
Outros
Sociedade

- Quais as entidades que comecem por F?

```
select e.designacao
FROM entidade e
WHERE e.designacao LIKE 'F%'
```

designacao
Freguesia de Carvalhosa
Freguesia de Joane
Freguesia de Moreira de Cónegos
FCC ENVIRONMENT PORTUGAL, S.A.
FILHOS LDA
Fresenius Medical Care Portugal, SA
Fidelidade Companhia de Seguros, S.A.
Freguesia de Ronfe
Freguesia de Avenidas Novas
Frineiva - Comercio de Produtos Alimentares, Lda.
Filhos S.A.

- **Soma dos Preços Contratuais por País?**

```
SELECT
p.nome AS pais,
SUM(c.precoContratual) AS total_preco
FROM contrato c
JOIN contrato_local_execucao cle
ON c.idContrato = cle.idContrato
JOIN municipio m
ON cle.municipio_id = m.municipio_id
JOIN distrito d
ON m.distrito_id = d.distrito_id
JOIN pais p
ON d.pais_id = p.pais_id
GROUP BY p.nome
```

pais	total_preco
Alemanha	192630.15
Arábia Saudita	103500
Bélgica	107779.45
Cabo Verde	8550
Emiratos Árabes Unidos	15995
Espanha	558392.09
Estados Unidos	716.11
França	118970
Hungria	1507.24
Indonésia	4711.82
Itália	5400
Panamá	7405

2. Médias

- **Quantos contratos existem por tipo de contrato?**

```
SELECT tc.nome,
COUNT(c.idContrato) AS quantidade
FROM contrato c
JOIN
ClassificacaoContratos ctc ON c.idContrato =
ctc.idContrato
JOIN
tipo_contrato tc ON ctc.tipoContrato_id =
tc.tipoContrato_id
```

GROUP BY tc.tipoContrato_id

nome	quantidade
Aquisição de serviços	9645
Empreitadas de obras públicas	1062
Aquisição de bens móveis	10930
Locação de bens móveis	236
Concessão de serviços públicos	22
Sociedade	8
Concessão de obras públicas	4
Outros	4

- **Total de Contratos por Entidade Adjudicante?**

```
SELECT e.designacao AS entidade,
COUNT(c.idContrato) AS total
FROM contrato c
JOIN
entidade e ON c.adjudicante_id = e.entidade_id
GROUP BY c.adjudicante_id
ORDER BY total DESC
```

entidade
Unidade Local de Saúde de Gaia/Espinho, E. P. E.
Unidade Local de Saúde de Santa Maria, E. P. E.
Centro Hospitalar Tondela-Viseu, E. P. E.
Águas do Norte, S. A.
Instituto Português de Oncologia de Lisboa Francisco Gentil, E. P. E.
Unidade Local de Saúde da Guarda, E. P. E.
Hospital Professor Doutor Fernando Fonseca, E. P. E.
Unidade Local de Saúde da Região de Leiria, E. P. E.
Unidade Local de Saúde do Nordeste, EPE
Unidade Local de Saúde do Alto Minho, E. P. E.
Centro Hospitalar Universitário Lisboa Central, E. P. E.

- **Média do preço contratual por tipo de procedimento?**

```
SELECT tp.descricao AS procedimento,
ROUND(AVG(c.precoContratual), 2) AS media_preco
FROM contrato c
JOIN tipo_procedimento tp ON
c.tipoProcedimento_id = tp.tipoProcedimento_id
GROUP BY tp.tipoProcedimento_id
```

procedimento	media_preco
Consulta Prévia	25780.99
Ajuste Direto Regime Geral	29701.03
Concurso público	251547.89
Ao abrigo de acordo-quadro (art.º 259.º)	86701.97
Ao abrigo de acordo-quadro (art.º 258.º)	60555.68
Consulta prévia ao abrigo do artigo 7º da Lei n.º 30/2021, de 21.05	149875.5
Consulta Prévia Simplificada	156074.64
Concurso limitado por prévia qualificação	1009613.6
Setores especiais – isenção parte II	229849.13
Contratação excluída II	32716.66
Concurso público simplificado	2568353.68

- **Número de contratos por mês?**

```
SELECT SUBSTR(c.dataCelebracaoContrato, 6, 2) AS
mes,
COUNT(*) AS total
FROM contrato c
GROUP BY mes
ORDER BY mes
```

mes	total
01	12849
02	8899

- **Contratos com preço superior à média global?**

```
WITH media_global AS (
SELECT AVG(precoContratual) AS mg FROM contrato
)
SELECT c.idContrato,
e.designacao AS entidade,
c.precoContratual
FROM contrato c
JOIN entidade e ON c.adjudicante_id =
e.entidade_id
WHERE c.precoContratual > (SELECT mg FROM
media_global)
ORDER BY c.precoContratual DESC
```

idContrato	entidade
10583932	Infraestruturas de Portugal, S. A.
10531808	Unidade Local de Saúde de Santa Maria, E. P. E.
10572398	Metro do Porto, S. A.
10549497	Município de Santo Tirso
10562529	Infraestruturas de Portugal, S. A.
10432537	Infraestruturas de Portugal, S. A.
10535059	Secretaria-Geral do Ministério da Administração Interna
10562545	Infraestruturas de Portugal, S. A.
10466367	MARINHA - Direção de Navios
10534616	Câmara Municipal de Matosinhos
10562550	Infraestruturas de Portugal S. A.

3. Difíceis

- **Top 10 contratos mais caros com cpv associado?**

```
SELECT c.idContrato,
c.precoContratual,
cpv.designacao AS cpv_nome
FROM contrato c
JOIN possui p ON c.idContrato = p.idContrato
JOIN cpv ON p.cpv_id = cpv.cpv_id
ORDER BY c.precoContratual DESC
LIMIT 10
```

idContrato	precoContratual	cpv_nome
10583932	44912900.05	Construção de auto-estradas
10531808	30636000	Produtos farmacêuticos
10572398	29523182.59	Material circulante
10549497	19549998.24	Serviços de recolha de resíduos
10562529	17684053.95	Serviços de manutenção de vias férreas
10432537	17127407.78	Reparação de estradas
10535059	13252420	Equipamento e material informático
10562545	12516859.7	Serviços de manutenção de vias férreas
10466367	12024740	Combustíveis
10534616	11973242.46	Obras de restauro

- **Top 10 municípios por soma do preço contratual?**

```
WITH contratos_por_municipio AS (  
  SELECT DISTINCT idContrato, municipio_id  
  FROM contrato_local_execucao  
)  
SELECT m.nome AS municipio,  
ROUND(SUM(c.precoContratual), 2) AS total  
FROM contratos_por_municipio cm  
JOIN contrato c ON cm.idContrato = c.idContrato  
JOIN municipio m ON cm.municipio_id =  
m.municipio_id  
GROUP BY m.municipio_id  
ORDER BY total DESC  
LIMIT 10
```

municipio	total
Lisboa	288070043.34
Desconhecido	122587771.68
Porto	121863804.38
Concelho não determinado	56000682.46
Vila Nova de Gaia	52195101.79
Santiago do Cacém	50718080.16
Faro	33561063.23
Coimbra	33478469.08
Santo Tirso	27482225.06
Almada	26757518.35

2.3. Aplicação Python

A componente final do projeto consiste num site, desenvolvido em Python, que serve como interface de consulta e visualização dos dados contidos na base de dados SQLite (contratos_publicos.db). Esta aplicação demonstra a funcionalidade analítica do modelo relacional através da execução interativa das interrogações SQL definidas.

1. Arquitetura da aplicação

O desenvolvimento recorreu ao Flask, reconhecido pela sua leveza e flexibilidade. A arquitetura da aplicação é dividida em três camadas essenciais:

- **Servidor:** (server.py) O ponto de entrada da aplicação. Este ficheiro é responsável por inicializar o ambiente de execução e estabelecer a **conexão inicial** com a base de dados antes de iniciar o servidor web.

```
#!/usr/bin/python3
import logging
from app import APP
import db

if __name__ == '__main__':
    logging.basicConfig(level=logging.INFO,
                        format='%(asctime)s - %(levelname)s - %(message)s',
                        datefmt='%Y-%m-%d %H:%M:%S')
    db.connect()
    APP.run(host='0.0.0.0', port=9000)
```

- **Lógica da Aplicação:** (app.py) Contém o código central, onde são definidas todas as rotas (*endpoints*) e as funções que executam a lógica. neste caso, as consultas SQL.
- **Apresentação:** (Templates HTML) Utiliza o motor de *templating* Jinja2 (integrado no Flask) para renderizar a estrutura HTML. Os dados obtidos das consultas SQL são passados a estes *templates* para serem formatados e apresentados ao utilizador.

2. Interface da app

- / (index.html)

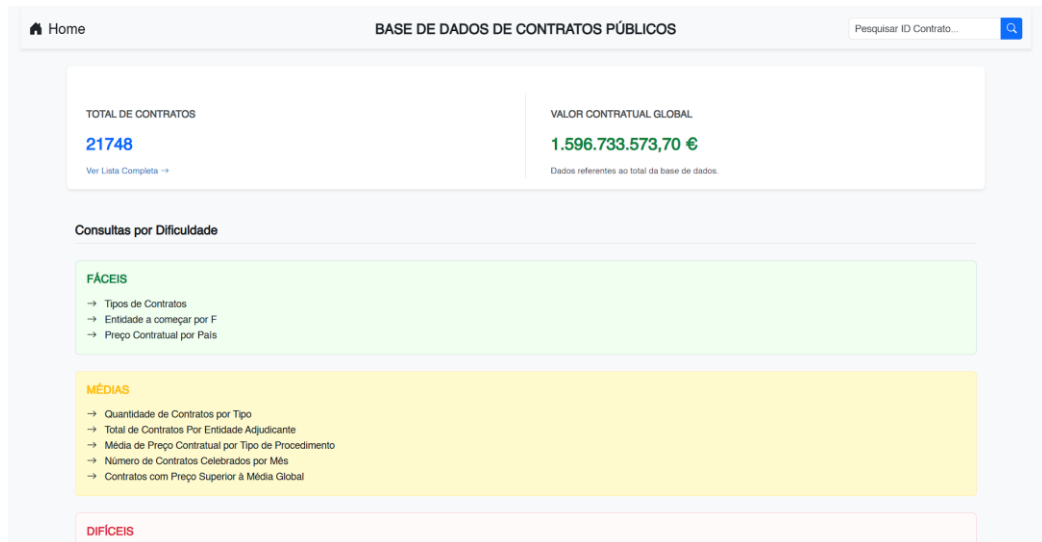
Página inicial da aplicação, de Título ‘Base de Dados de Contratos Públicos’, com informação sobre o número total de contratos e valor total da soma de todos os contratos, uma opção para listar todos os contratos (/contratos.html), podendo depois obter informação sobre um contrato apenas clicando no seu ID (/contrato.html), ou na página inicial no canto superior direito indicar o idContrato e ser logo direcionado para o mesmo.

```
@APP.route('/contratos/')
def listar_contratos():
    rows = db.execute('''
        SELECT idContrato, objetoContrato, precoContratual, dataCelebracaoContrato
        FROM contrato
        ORDER BY dataCelebracaoContrato DESC
    ''').fetchall()
    return render_template('contratos.html', contratos=rows)

@APP.route('/contratos/<int:idcontrato>/')
def contrato(idcontrato):
    contrato = db.execute('''
        SELECT *
        FROM contrato
        WHERE idcontrato = ?
    ''', (idcontrato,)).fetchone()
    return render_template('contrato.html', c=contrato, voltar=request.referrer)
```

Código para listar todos os contratos e informação sobre um contrato em específico.

(index.html)



- / (Q1 ... /Q10.html)
No index.html temos acesso a 10 perguntas de dificuldades variadas que, escolhendo uma, direciona para uma nova página com a pergunta e o resultado tabelado.

Exemplo da pergunta Q6

[Home](#)

Média do Preço Contratual por Tipo de Procedimento

Procedimento	Média (€)
Consulta Prévia	25780.99
Ajuste Direto Regime Geral	29701.03
Concurso público	251547.89
Ao abrigo de acordo-quadro (art.º 259.º)	86701.97
Ao abrigo de acordo-quadro (art.º 258.º)	60555.68
Consulta prévia ao abrigo do artigo 7º da Lei n.º 30/2021, de 21.05	149875.5
Consulta Prévia Simplificada	156074.64
Concurso limitado por prévia qualificação	1009613.6
Setores especiais – isenção parte II	229849.13
Contratação excluída II	32716.66
Concurso público simplificado	2568353.68

Código .html para a pergunta:

```
{% extends 'base.html' %}
{% block content %}
<h1>Média do Preço Contratual por Tipo de Procedimento</h1>
<table border="1">
  <tr>
    <th>Procedimento</th>
    <th>Média (€)</th>
  </tr>
  {% for d in dados %}
  <tr>
    <td>{{ d['procedimento'] }}</td>
    <td>{{ d['media_preco'] }}</td>
  </tr>
  {% endfor %}
</table>
{% endblock %}
```

“Endpoint”	Funcionalidade
/	Página de entrada com acesso a número total de contratos assim como todos os contratos, e lista de perguntas
/contratos	Tabela com todos os contratos
/contrato	Tabela de um contrato só, previamente selecionado
/Q1.../Q10	Tabela com a resposta para as perguntas selecionadas

3. Conclusão

Para o trabalho realizado foi feita uma compreensão sólida e sistemática no desenvolvimento de um sistema de bases de dados para armazenar e consultar informações sobre Contratos Públicos em Portugal 2024.

O projeto começou com uma análise detalhada dos requisitos, na criação de um Modelo Entidade-Relacionamento (ER), que foi então rigorosamente mapeado para um Modelo Relacional.

A decisão de desagregar a tabela única original em entidades como *Contrato*, *Entidade*, *CPV*, *Localização* e *Acordo Quadro* foi crucial para a correta normalização. O Modelo Relacional final está em conformidade com a Terceira Forma Normal (3FN), garantindo a integridade e a ausência de dependências transitivas e parciais.

A complexidade foi gerenciada com a utilização da biblioteca pandas para a extração e o desenvolvimento de funções de parsing avançadas em Python para lidar com dados não estruturados, como *adjudicatários* e *fundamentação*.

A eficácia do modelo relacional foi comprovada pela capacidade de executar interrogações SQL, divididas por níveis de dificuldade (Fáceis, Médias e Difíceis), que abrangem desde consultas simples até análises mais complexas.

A componente final, o site desenvolvido em Python usando Flask, proporciona uma interface de consulta e visualização dos dados, tornando o projeto funcional e acessível ao utilizador.

Em última instância, decidimos deixar nosso projecto disponível no **Github** para eventuais consultas e futuras melhorias (caso sejam necessárias):

<https://github.com/ezequielcabeja/BD-Projecto-Contratos-Publicos-2024>

Referências

Ferramentas:

Link da ferramenta (**dbdia**) usada para ilustração dos diagramas do modelo ER:

<https://colab.research.google.com/github/edrdo/dbdia/blob/master/src/main/colab/dbdia.ipynb>

Link da ferramenta (**draw.io**) usada para ilustração dos diagramas do modelo Relacional:

<https://app.diagrams.net/?src=about>

Slides das aulas:

O **Modelo ER** - Bases de Dados (CC2005), *Eduardo R. B. Marques, DCC/FCUP*

O **Modelo Relacional** - Bases de Dados (CC2005), *Eduardo R. B. Marques, DCC/FCUP*

Normalização - Bases de Dados (CC2005), Departamento de Ciência de Computadores

Faculdade de Ciências da Universidade do Porto, *Eduardo R. B. Marques — DCC/FCUP*

SQL Gabriel - Bases de Dados, *Gabriel David, gtd@fe.up.pt*

Aplicações BD com SQL embebido - Bases de Dados (CC2005), *Eduardo R. B. Marques, DCC/FCUP*