# HelpLineDB – Scripts e Documentação

Este diretório consolida os scripts do banco (SQL Server 2019) e instruções para verificação e geração de diagramas.

## Arquivos

- `helplinedb_full.sql` : script completo (idempotente) para criar o banco HelpLineDB, tabelas, FKs, índices, procedures, view e (opcional) login/GRANTs.
- `helplinedb_checks.sql` : consultas de verificação (tabelas, índices, FKs, rotinas, view) para confirmar o estado do BD.

## Como aplicar (SSMS)

1. Abra uma nova query e cole o conteúdo de `helplinedb_full.sql` .
2. Execute tudo de uma vez (F5). Os blocos `GO` separam os batches.
3. Rode `helplinedb_checks.sql` para validar.

Observações:

- Se a "Lista de Erros" acusar objetos já existentes, mas a janela "Mensagens" indicar sucesso, atualize o IntelliSense (Ctrl+Shift+R) e reabra a query (o cache do SSMS pode ficar desatualizado).
- O bloco de criação de login/usuário de aplicação está comentado; ajuste a senha e descomente se desejar criar o usuário `helpline_app` com GRANTs mínimos.

## Smoke Test rápido

Após criar o BD, você pode testar:

1. Inserir um usuário de teste ( `admin@helpline.local` ).
2. Executar `sp_Tickets_Create` para gerar um protocolo `HL-YYYYMMDD-XXX` .
3. Adicionar uma mensagem ao ticket com `sp_TicketMessages_Add` .
4. Rodar `sp_Reports_Tickets` e consultar a view `v_Tickets_Daily_Aggregates` .

Use o arquivo `helplinedb_checks.sql` para facilitar.

## Como gerar diagramas (MER/DER/ERD)

### 1) SSMS – Database Diagrams (nativo)

1. No SSMS, expanda seu banco `HelpLineDB` .
2. Clique com o botão direito em "Database Diagrams" → "New Database Diagram".
3. Se for a primeira vez, o SSMS pedirá para instalar os objetos de diagrama – confirme.
4. Selecione todas as tabelas (User, Role, UserRole, Ticket, TicketCategory, TicketLevel, TicketPriority, TicketStatus, TicketSequence, TicketMessage) e clique em "Add".
5. Ajuste o layout e salve o diagrama.

### 2) SSMS – View Dependencies

1. Clique com o botão direito na tabela `Ticket` → "View Dependencies".
2. Você verá relações com `TicketCategory` , `TicketLevel` , `TicketPriority` , `TicketStatus` e `User` .

### 3) Azure Data Studio (alternativa)

- Instale extensões de visualização de esquema (por exemplo, "Schema Visualization").

- Conecte no SQL Server, abra o banco e gere o diagrama pelas extensões.

### 4) Exportar modelo externo (dbdiagram.io / Draw.io / Mermaid)

1. Gere um "esqueleto" com colunas e FKs via scripts (query abaixo) e cole em ferramentas como dbdiagram.io (DBML) ou use Mermaid ER.

Exemplo de consulta de FKs para mapear relacionamentos:

```
SELECT
  pt = OBJECT_NAME(fk.parent_object_id), pc = cp.name,
  rt = OBJECT_NAME(fk.referenced_object_id), rc = cr.name,
  fk.name AS fk_name
FROM sys.foreign_keys fk
JOIN sys.foreign_key_columns fkc ON fkc.constraint_object_id = fk.object_id
JOIN sys.columns cp ON cp.object_id = fkc.parent_object_id AND cp.column_id =
fkc.parent_column_id
JOIN sys.columns cr ON cr.object_id = fkc.referenced_object_id AND cr.column_id =
fkc.referenced_column_id
WHERE fk.parent_object_id IN (OBJECT_ID('dbo.Ticket'), OBJECT_ID('dbo.UserRole'),
OBJECT_ID('dbo.TicketMessage'))
ORDER BY pt, fk.name;
```

## Padrões adotados

- Datas em UTC ( `SYSUTCDATETIME()` ), chaves `GUID` com `NEWSEQUENTIALID()` para `User` e `Ticket` .
- Protocolo gerado por dia via `TicketSequence` e `sp_Tickets_Create` (atômico com `UPDLOCK/HOLDLOCK` ).
- Mensagens por ticket em `TicketMessage` ("chat" interno com remetentes: user/analyst/bot/sistema).
- Índices para filas (status/assignee) e relatórios.

## Próximos passos

- Backend (ASP.NET Core recomendado): conectar no `HelpLineDB` e expor rotas:
  - `GET /health`
  - `POST /tickets` (usa `sp_Tickets_Create` )
  - `POST /tickets/{id}/messages` (usa `sp_TicketMessages_Add` )
  - `GET /reports` (usa `sp_Reports_Tickets` )
  - `POST /chat` (proxy para LLM no servidor; NUNCA expor a chave no app)