Para criar a nossa base de dados optamos por um container de forma a poupar espaço em disco. Criamos um docomento Docker-compose, onde incluímos a imagem do SQL Server 2019 na ultima versão, o nome do container (sql\_server), a porta não sera mapeada para uma porta diferente da 1433, e os volumes de backup, persistência de dados e logs. Decidimos ainda criar a password num ficheiro separado para evitar que esta fosse diretamente no nosso Docker-compose.

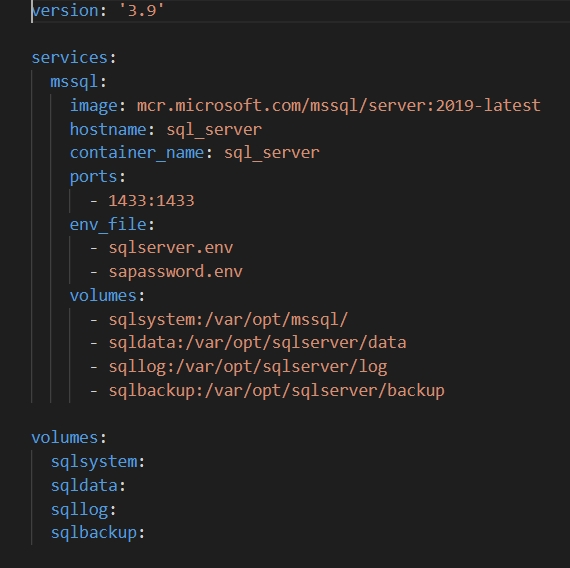


Figura 1 - Docker-Compose SQL

Depois de correr o comando Docker-compose up e nos certificarmos que o container está a correr, podemos aceder via SQL Manager à nossa Base de dados. Para isso colocamos o IP da maquina onde o container está a correr, neste caso será o nosso localhost, a porta é irrelevante, uma vez que a mesma não foi alterada da porta padrão 1433, no entanto se tivéssemos alterado, teríamos que indicar da seguinte forma.

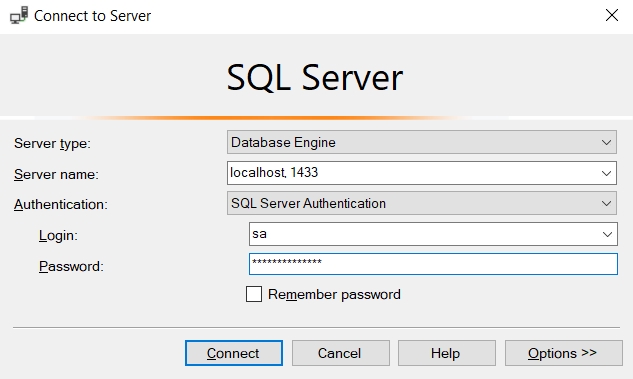
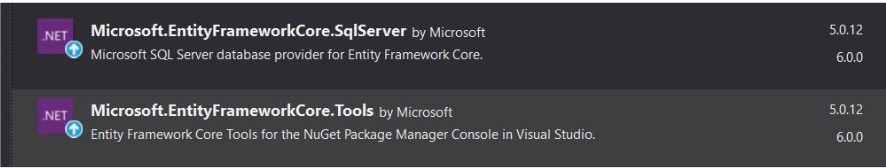


Figura 2 - SQL Manager Login

Antes de podermos criar o nosso código para inserção das tabelas, devemos criar a base de dados que será utilizada. Vamos criar um base de dados chamada Banco.

A partir daqui vamos abordar uma metodologia “code first” para nos certificarmos que as alterações ao nosso código serão reproduzidas facilmente na base de dados.

Começamos por instalar dois pacotes no nuGet Manager. Optamos pela versão 5.0.12 para evitar erros de compatibilidade.



Para organizar melhor o nosso código, vamos criar uma pasta database, onde criaremos uma classe de ligação chamada BankContext, esta classe vai herdar da classe DbContext que veio dos nossos pacotes instalados. Nesta classe vamos criar DBSets para as nossas classes que serão usados posteriormente para as operações CRUD. Criamos ainda a nossa ligação, fazendo override do método onConfiguring. Novamente aqui a porta será irrelevante uma vez que é a porta padrão, decidimos colocar para que assim se houver alteração de porta poderá ser mais fácil visualmente mudar. Devemos ainda colocar o nome da Base de dados criada, o utilizador (deverá ter acesso à mesma) e a sua password. A partir daqui estamos prontos para começar as migrações. Uma chamada de atenção, nas classes, o Id que será usado na tabela deverá conter o nome exato da classe + id,(classe Cliente = ClienteId), para que o Visual Studio consiga perceber que esta é a Primary Key da nossa tabela. Tambem fizemos algumas alterações nos nossos modelos para que na migração não fossem criados nvarchar de tamanha máximo na base de dados, bem como colocar algumas propriedades como “required” para que não possam ser nulas.

Uma imagem com texto

Descrição gerada automaticamente

Figura - Tamanho Varchar

Por fim só temos que criar a migração e fazer update da base de dados.

Antes de fazer Update devemos confirmar no ficheiro criado se as tabelas serão criadas como pretendemos, e só depois fazer Update-database.

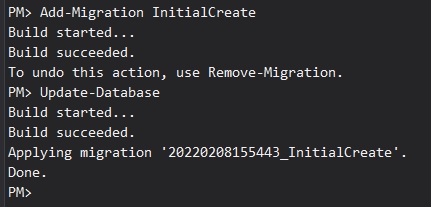


Figura 4 - Migrations iniciais

Se formos ver a nossa base de dados já teremos as tabelas de acordo com o nosso código.

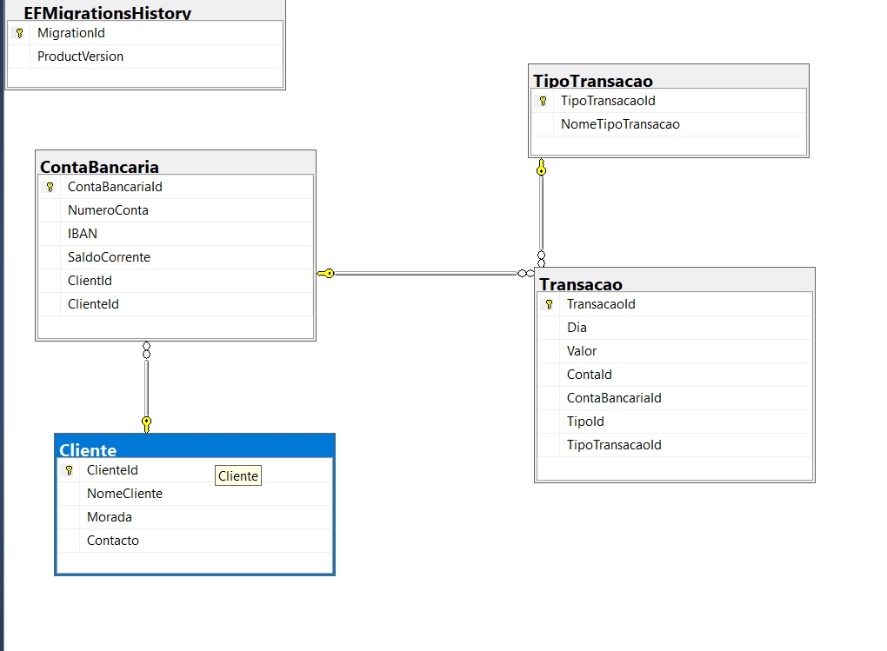


Figura 5 - Diagrama DB

As colunas nvarchar também não foram criadas com tamanho máximo, como definido.

Uma imagem com mesa

Descrição gerada automaticamente

Figura - Design Cliente

Para podermos ter aguns dados, vamos criar “Seeds” que vão popular a nossa Base de dados, para manter o código organizado vamos criar na pasta de Databse um ficheiro chamado Configurations, onde teremos as configurações para cada modelo. Estas classes iram herdar da interface “IEntityTypeConfiguration” e daremos override do método “Configure”. Indicamos a tabela e criamos os objetos a inserir. Devemos certificar-nos de respeitar as “Foreign Keys”.

Uma imagem com texto

Descrição gerada automaticamente

Figura – Seeds

Na nossa classe “BankContext” adicionamos o método “OnModelCreations” onde chamamos estas classes.

Uma imagem com texto

Descrição gerada automaticamente

Figura – OnModelCreation

Criamos uma nova migração, certificamo-nos que está a criar corretamente o que queremos e fazemos update. Se voltarmos à nossa base de dados, podemos confirmar que a mesma já contempla os dados que inseremos nas classes de configuração.

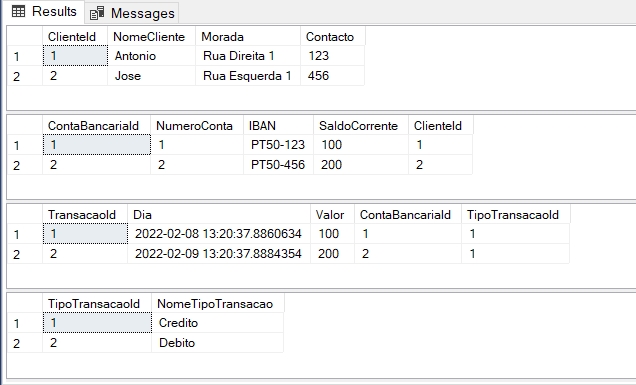


Figura - Tabelas com dados