Gestão de Gabinete - Padrões do Projeto

# Arquitetura de Tenant (Tenancy)

O sistema “gestão de gabinete”, desde sua primeira versão, foi criado para atender múltiplos usuários, no começo o foco era os gabinetes da câmara municipal de Bauru, e posteriormente disponibilizado para qualquer câmara que queira ter acesso ao sistema, com isto desde sua criação, precisou preocupar com a arquitetura de software em questão de usuários e banco de dados.

## Single tenant

A primeira arquitetura adota foi a single *tenant¹* ou *tenancy*, ambas tem o mesmo significado, que consiste em uma instância do sistema com sua *database* isolada para cada usuário.

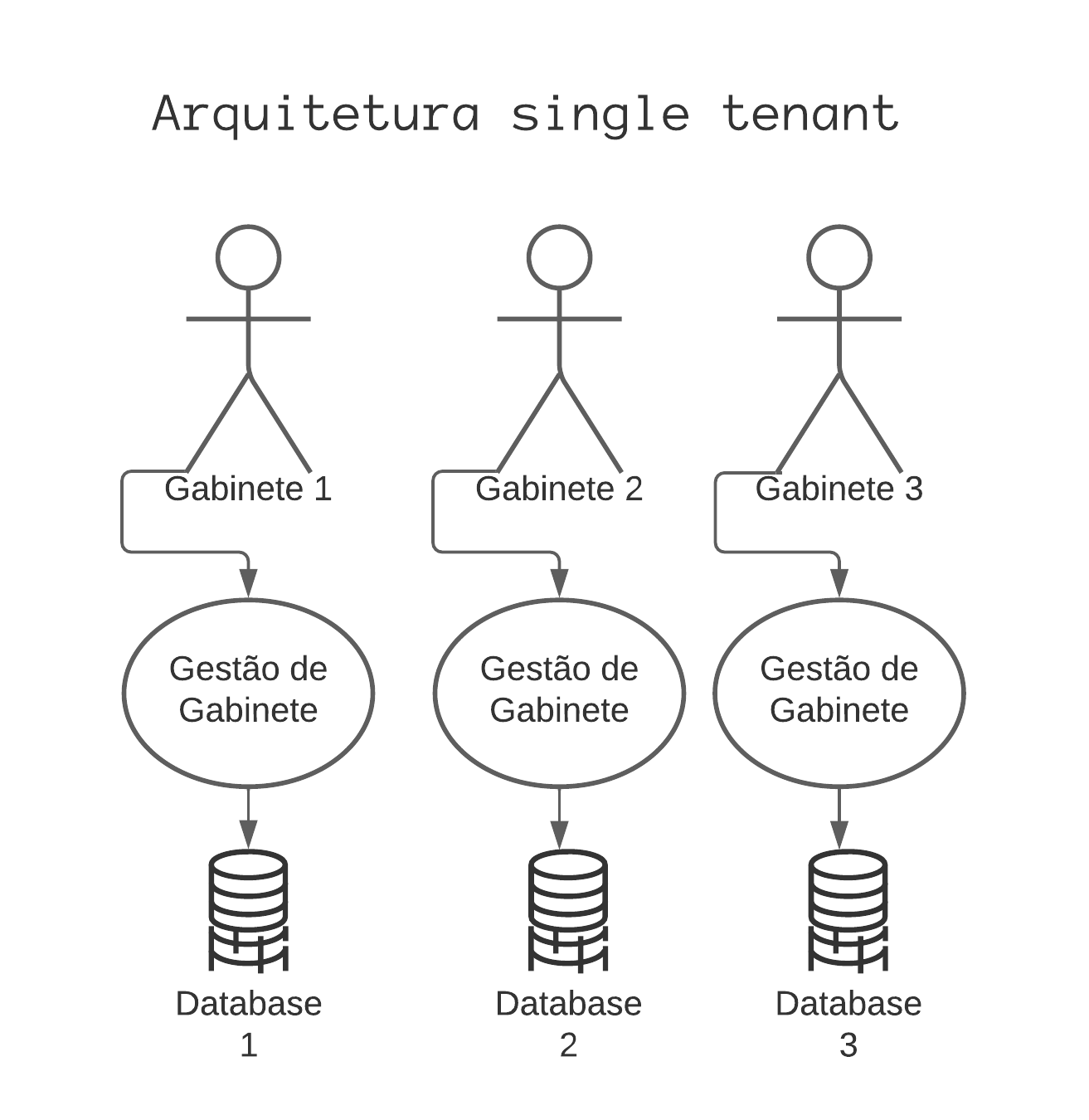


Figura : Arquitetura single tenant.

Usando a câmara municipal de Bauru, para um exemplo da arquitetura, podemos pensar que existiam 17 instâncias do gabinete no servidor, ou seja, cada uma destas instâncias seria uma pasta no servidor, com uma instalação completa do sistema, e conectada a sua própria *database*.

Nesta versão se possui as vantagens que os dados são isolados de forma segura, uma aplicação só tem acesso ao seu banco de dado. Entretanto, possui as desvantagens que cada versão do sistema precisa ser atualizada individualmente, além de dificultar escalar a aplicação.

Foi para corrigir estes problemas, que foi pensando em uma arquitetura multi-tenant/tenancy.

## Multi tenant

Em uma arquitetura de multi tenant, como o nome da à dica, consiste em vários inquilinos para um mesmo sistema. No nosso contexto, seriam vários gabinetes acessando uma mesma instancia[[1]](#footnote-1) do “gestão de gabinete”. Neste caso precisaríamos de apenas uma instalação do software e poderíamos cadastrar quantos gabinetes quiséssemos.

Existem duas versões desta arquitetura: single e multi database.

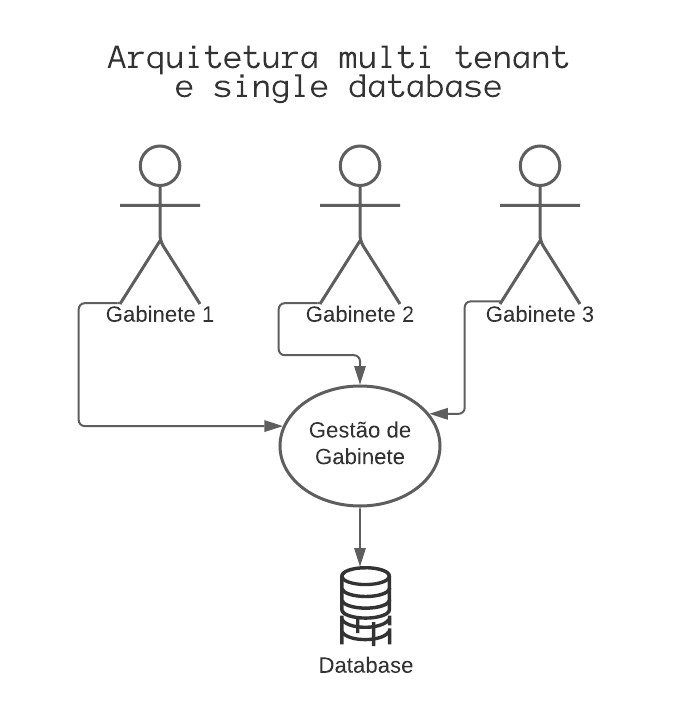
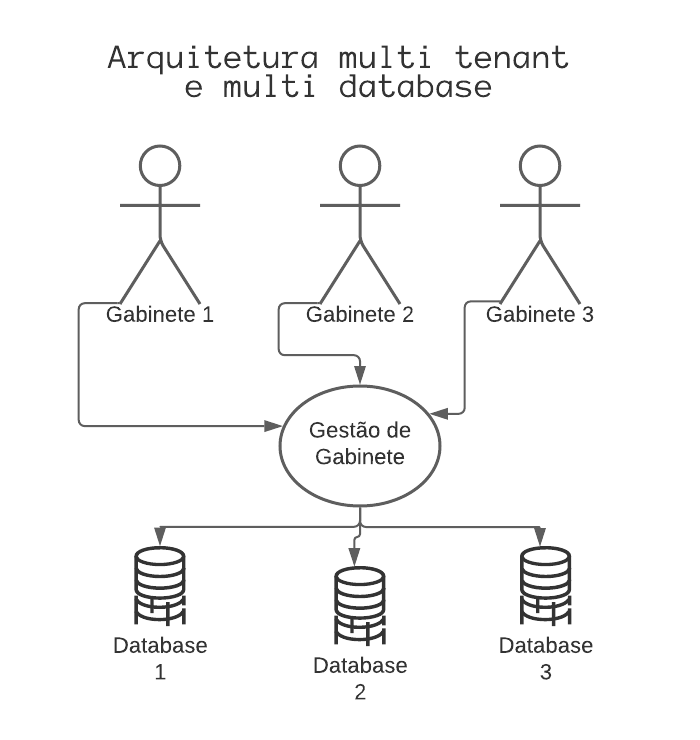
 

Figura 3: Arquitetura multi tenant - multi database.

Figura 2: Arquitetura multi tenant - single database.

Na arquitetura single database, temos o único banco de dados, com todos os cadastros (pessoa, documentos, atendimentos e etc...) sendo feito no mesmo. Diferenciasse quais os dados de cada gabinete por um ***campo\_id*** em comum do usuário e das tabelas do sistema.

Embora um banco de dados seja feito para aguentar uma grande quantia de registro, acreditamos que este modelo tenha uma manutenção mais difícil, além de uma escalabilidade, embora grande, limitada. Então a arquitetura escolhida foi o multi database. Nesta arquitetura existe apenas uma única instância do sistema, porém cada gabinete tem seu próprio banco de dados.

As vantagens desta ultima versão, é um sistema escalável, embora o multi-database possa exigir muito servidor em um nível grande de gabinetes, o próprio projeto deixa você conectar cada gabinete ao seu banco de dados, e com isto você pode ter alguns usuários em um banco local e outros em um banco na nuvem, por exemplo. Além disto, fica fácil a atualização, manutenção e se preserva integridade dos dados.

## Implementação do multi tenant

Existem diversas formas de se programar a arquitetura multi tenant, a escolhida por nós foi criar a conexão através de um campo de domínio na tabela de usuários.

No laravel existe um arquivo .***env*** em que temos configuradas as informações de acesso ao banco de dados. Na verdade o .env passa as informação para ***database.php*** e nele é que é realizado a conexão.

Para passar os dados da conexão, foi criada uma tabela chamada *organizacoes,* ela tem como campo o nome do gabinete, seu domínio e as configurações de acesso ao banco de dados (database, porta, host, password). Na tabela *user* também foi adicionada uma coluna de domínio chamada “domain”.

O sistema começa conectando a uma database padrão configurada no arquivo **.env** e nesta base de dados que rodamos a migração[[2]](#footnote-2) para criar a tabela de *organizacoes* e *user*, também podemos rodar o seed³ para ter um primeiro usuário cadastrado.

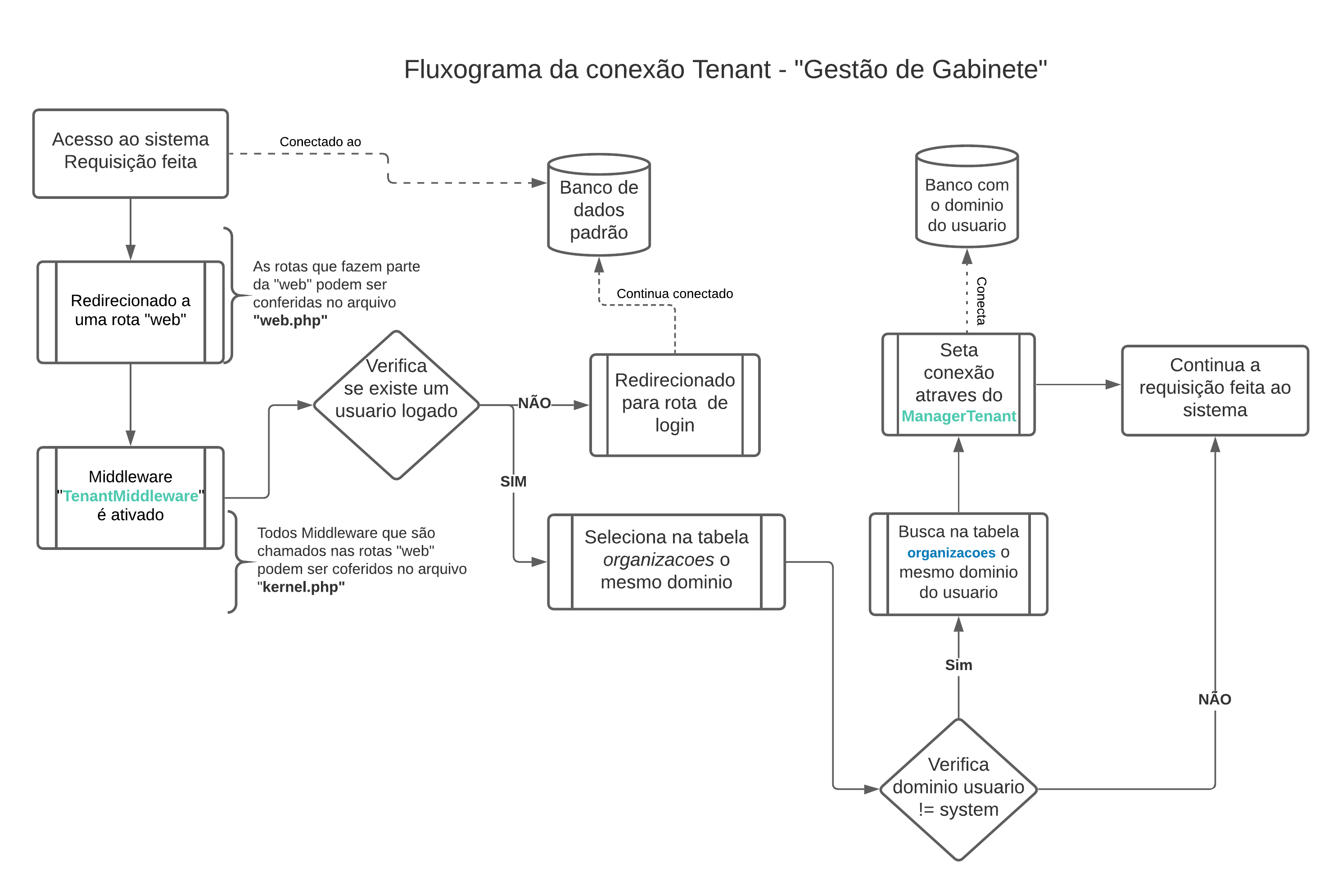


Figura 4: Fluxograma da conexão Tenant – “Gestão de Gabinete”

O sistema já esta trabalhando com múltiplos inquilinos a partir do momento que você acessa um url do sistema. Quem realiza as alterações do banco de dados é um Middleware chamado **TenantMiddleware.php**, ele é cadastrado para ser ativado em todas rotas “web” do sistema, isto acontece no arquivo **kernel.php**. Para saber quais rotas do sistema pertencem ao grupo “web” basta acessar o arquivo **web.php**.

O Tenant Middleware verifica se existe um usuário “logado”, caso não exista, continua as requisições normalmente e o usuário é enviado à tela de “login”. Caso exista e o domínio dele for diferente de “system” então vai se alterar as configurações do banco de dados. Para isto o sistema procura na tabela *organizações* uma que possua o domínio igual a do usuário e a partir dos dados desta tabela que vai “setar” a nova configuração do banco de dados, realizada pelo arquivo **ManagerTenant.php** no método *setConnection*.

A partir dai para qualquer rota “web” acessada pelo sistema este processo se repete com Tenant Middleware configurando o banco de dados correto para cada usuário.

1. :*Tenant* vem do inglês que significa “*inquilinos*” [↑](#footnote-ref-1)
2. *php artisan migrate //Os arquivos que serão executados estão na raiz da pasta “migrations”*

   ³ *php artisan db:seed//Os seeds que serão chamados estão no arquivo DatabaseSeeder.php* [↑](#footnote-ref-2)