

**Técnico de Informática – Sistemas 2019/2022**

2021/2022

**Prova de Aptidão Profissional**

GESRES 1.0

**Vítor Pereira Albernaz**

Ponta Delgada, março de dois mil e vinte e dois



**Técnico de Informática – Sistemas 2019/2022**

2021/2022

**Prova de Aptidão Profissional**

GESRES 1.0

**Formando:** Vítor Albernaz

**Orientador:** Duarte Cota

Ponta Delgada, março de dois mil e vinte e dois

**AGRADECIMENTOS**

|  |
| --- |
| Primeiramente, queria agradecer não só à instituição da Escola de Novas Tecnologias dos Açores pela oportunidade de tirar o meu curso de nível IV nesta instituição, mas também a todos os formadores e professores que pelo meu progresso formativo passaram. Da área técnica, destaco os formadores Jorge Soares; Jorge Costa; Luís Carreira; Daniel Oliveira; Nuno Melo; Duarte Cota e Óscar Neto, enquanto na área sociocultural e científica, saliento os professores(as) Pedro Fonseca; Paulo Martinho; Mariana Silva; Henrique Medeiros; Renato Lameirão; Silvia Gouveia; Ana Pereira e Anabela Ferreira. Também agradeço a todo o trabalho do *staff* da escola, a dona Gabriela Viveiros, a doutora Patrícia Ribeiro e o doutor João Lima, que sempre fizeram o melhor possível pela escola.  Um especial obrigado quer ao formador Duarte Cota, que me ajudou sempre que precisava enquanto orientador da minha prova de aptidão profissional, quer à diretora de turma, Ana Pereira, por nos ajudar no processo de conseguir estágio e de nos preparar para da melhor maneira para as nossas provas finais de curso.  Finalmente, gostaria de agradecer à minha turma por toda a convivência durante estes incríveis três anos do curso.  Obrigado pela confiança! |

Assinatura

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Resumo:**

GESRES é um software principalmente de vendas num estabelecimento bar ou restaurante, o qual, por sua vez, exerce alguns extras, tais como ligação a um programa para a cozinha, gestão de *stock* e de faturas em tempo real, e controlo de funcionários.

Para o desenvolvimento do projeto foram utilizados *softwares* com o fim de criar ambientes gráficos (SceneBuilder), ajudar e agilizar no processo de desenvolver a parte lógica da aplicação através do IDE Intellij (Ambiente de desenvolvimento integrado), utilizando a linguagem de programação JAVA com componentes do JavaFX SDK 17 e outras bibliotecas como mysql-connector para efetuar a conexão com a base de dados e *software*. Para a criação da base de dados e pesquisa na mesma, foi utilizada a linguagem SQl na aplicação MySQl Workbench.

//Resultados finais

//conclusão

Índice geral:

[Índice geral 6](#_Toc95690049)

[Índice de figuras 7](#_Toc95690050)

[Introdução 9](#_Toc95690051)

[Desenvolvimento 10](#_Toc95690052)

[Vista inicial- login 10](#_Toc95690053)

[Dashboards 11](#_Toc95690054)

[Dashboard- Admin 11](#_Toc95690055)

[Dashboard- Funcionário 16](#_Toc95690056)

[Dashboard- Cozinha 21](#_Toc95690057)

[Organização do projeto 23](#_Toc95690058)

[Softwares 23](#_Toc95690059)

[Otimização do código 24](#_Toc95690060)

[Organização dos ficheiros 25](#_Toc95690061)

[Base de dados 27](#_Toc95690062)

[Estrutura base de dados 28](#_Toc95690063)

[Conclusão 34](#_Toc95690064)

[Bibliografia 35](#_Toc95690065)

[Anexos 36](#_Toc95690066)

Índice de figuras:

[Figura 1 - Vista login 11](#_Toc95689981)

[Figura 2 - Vista Admin 12](#_Toc95689982)

[Figura 3 - Vista stock 14](#_Toc95689983)

[Figura 4 - Vista adicionar produto 15](#_Toc95689984)

[Figura 5 - Vista faturas 16](#_Toc95689985)

[Figura 6 - Vista mesas 17](#_Toc95689986)

[Figura 7 - Vista detalhes mesas 18](#_Toc95689987)

[Figura 8 - Vista pagamento 18](#_Toc95689988)

[Figura 9 - Vista código funcionário 19](#_Toc95689989)

[Figura 10 - Vista pedidos 20](#_Toc95689990)

[Figura 11 - Vista cozinha 22](#_Toc95689991)

[Figura 12 - Bibliotecas JAVAFX 23](#_Toc95689992)

[Figura 13 - Biblioteca MySQl-connector 23](#_Toc95689993)

[Figura 14 - Exemplo função classe "MySQlConnection.java" 24](#_Toc95689994)

[Figura 15 - Import classe "MySQlConnection.java" 24](#_Toc95689995)

[Figura 16 - Retorno função da classe "MySQlConnection.java" 24](#_Toc95689996)

[Figura 17 - Arquitetura MVC 25](#_Toc95689997)

[Figura 18 - Pastas projeto MVC 26](#_Toc95689998)

[Figura 19 - Pasta "controller" 26](#_Toc95689999)

[Figura 20 - Pasta "model" 26](#_Toc95690000)

[Figura 21 - Pasta "view" 26](#_Toc95690001)

[Figura 22 - Ligação à base de dados 27](#_Toc95690002)

[Figura 23 - Modelo base de dados 28](#_Toc95690003)

[Figura 24 - Tabela "fornecedor" 29](#_Toc95690004)

[Figura 25 - Tabela "tipo" 29](#_Toc95690005)

[Figura 26 - Tabela "produto" 30](#_Toc95690006)

[Figura 27 - Tabela "funcionario" 31](#_Toc95690007)

[Figura 28 - Tabela "mesas" 31](#_Toc95690008)

[Figura 29 - Tabela "pedidos" 32](#_Toc95690009)

[Figura 30 - Tabela "detalhespedidos" 32](#_Toc95690010)

[Figura 31 - Tabela "tipospagamento" 33](#_Toc95690011)

[Figura 32 - Tabela "faturas" 33](#_Toc95690012)

Introdução

GESRES é um programa de vendas e gestão de um estabelecimento de bar e ou restauração.

Este projeto tem como objetivos, por um lado, a aplicação de conhecimentos obtidos nas aulas e/ou pesquisas na área da programação com interfaces gráficas, e gestão de bases de dados e, por outro, a construção de uma ferramenta de fácil utilização e aprendizagem da mesma. Além disso, pretende também permitir que a mesma consiga gerir pedidos entre o setor de Bar e Cozinha, integrar um sistema de gestão de stock através dos pedidos dos clientes e gerir a origem dos pedidos (estabelecimento, *takeaway* ou entregas ao domicílio, se existirem).

Um programa como o referido tem muita importância no mundo dos negócios, pois pode facilitar em muito tanto o trabalho dos funcionários como até mesmo o do administrador do estabelecimento.

Escolhi este tema por ser uma realidade de trabalho bastante comum na sociedade atual, sendo que o desenvolvimento do mesmo me desafiou no sentido de perceber o melhor e o mais eficaz para um utilizador em período de trabalho. O meu maior objetivo - criar uma interface onde seja de fácil compreensão e aprendizagem das suas funções - fez despertar mais em mim o gosto por desenvolver aplicações deste género, seja na parte do f*ront end,* seja no *back end*.

Desenvolvimento

Inicialmente, comecei por desenhar o programa, o qual ia ser com três dashboards, “Admin”, “Funcionário” e “Cozinha”, porém, decidi separar a dashboard da cozinha para um programa separado. Posteriormente, realizei as vistas do programa com a ajuda do SceneBuilder, ferramenta que gerou o código FXML para uma interface gráfica no meu programa. Ao todo, foram criadas quinze vistas para o programa:

-Uma de login;

-Sete dedicadas à dashboard de “Admin”;

-Sete dedicadas à dashboard de “Funcionário”;

-Uma para o programa da “Cozinha”;

Todas as vistas foram feitas para uma medida de aproximadamente, ou não exceder, as 15 polegadas, medida vulgarmente utilizada nos monitores de estabelecimentos do tipo bar/restaurante.

Vista inicial- login

Vista composta por uma ComboBox, onde os seus valores são os *users* disponíveis para o programa, no caso “Admin” e “Funcionário”, e por uma PasswordField para colocar a respetiva palavra-passe.

Esta tem como objetivo direcionar o utilizador, através de credenciais de acesso, para o destino desejado, sendo, igualmente, uma forma de guardar informações restritas expostas na dashboard de “Admin”.



Figura - Vista login

Dashboards

Dashboard- Admin

A principal de “Admin” foi elaborada para adicionar novos funcionários e removê-los, caso necessário, e para controlar a atividade dos mesmos, onde temos acesso a opções expostas no MenuBar para controlo de *stock*, receitas e faturas, informações estas que um empregado normal não terá acesso sem as credenciais de acesso.

Logo ao entrar na dashboard, deparamo-nos com três TextFields (para o primeiro nome, último nome e número de funcionário), um DatePicker (para a data de nascimento) e uma ComboBox (para selecionar o sexo do funcionário). Estes servem para adicionar um funcionário após carregar no botão “Adicionar”, cuja função é preparar um objeto da classe “Funcionarios” para ser introduzido na base de dados.

Pode encontrar-se também uma tabela, onde se apresentam alguns dados do funcionário: nome completo, atividade e número de funcionário. Esta é preenchida logo ao entrar na vista através do método “initialize()”, que preenche a mesma com os respetivos valores presentes na base de dados.

A atividade é passível de ser mudada através de uma a ComboBox e do botão “Atualizar” abaixo da tabela, que atualiza na base de dados a atividade através de um querry SQl de “UPDATE”, e, seguidamente, atualiza a tabela limpando todos os dados e voltando a colocá-los.



Figura - Vista Admin

No MenuBar presente, há dois menus: “STOCK” (com dois submenus “Consultar Stock” e “Adicionar Produto”) e “FATURAS” (com três submenus “Diárias”, “Semanais” e “Mensais”.)

//foto menu bar com opções todas abertas

O primeiro submenu, “Consultar Stock”, tem como objetivo possibilitar ao chefe ou o patrão do estabelecimento ir verificando e atualizando o *stock* disponível.

Ao carregar no mencionado submenu é aberta outra vista composta por uma tabela com o nome de cada produto e a respetiva quantidade em *stock*. Esta informação também é posta ao iniciar a vista através do método “initialize()”, que, através de um querry SQl, apresenta a lista de produtos presente na base de dados, sendo guardada num objeto da classe “Produtos”.

Para editar o *stock* de um produto nessa vista, seleciona-se o produto que se pretende editar no *stock* na tabela; quando este é selecionado, surge o nome do produto na primeira TextField e fica outra TextField para o utilizador poder colocar a nova quantidade de *stock*. Após carregar no botão “Editar”, é atualizada a informação da tabela limpando todos os dados e voltando a colocá-los.

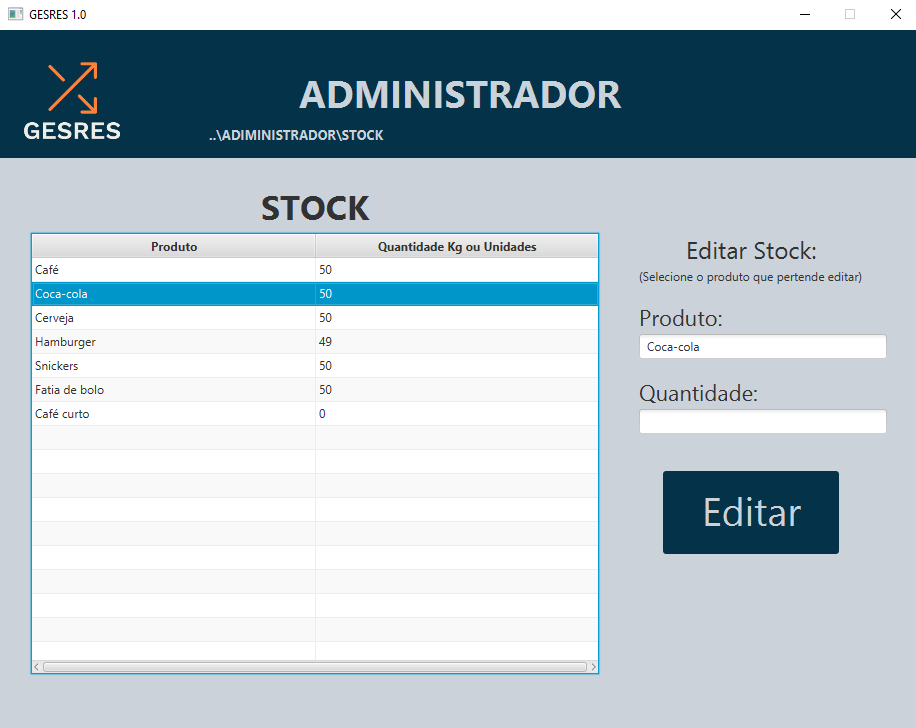


Figura - Vista stock

Quanto ao segundo submenu, “Adicionar Produto”, este tem com o objetivo adicionar produtos à base de dados para serem disponibilizados para venda.

Ao carregar no mesmo, é aberta outra vista composta por duas TextFields e duas ComboBoxs, as quais têm como função pôr o nome do produto a querer adicionar e o preço, selecionar o fornecedor e o tipo do produto, e identificar se se trata de “Café”, “Sumos”,” Bebidas Alcoólicas”, “Cozinha”, “Snacks”, “Doces/Sobremesas” ou “Retalho”. Após carregar no botão “Concluir”, os dados que são recolhidos dos quatro elementos são guardados num objeto da classe “Produtos” e, de seguida, adicionados à base de dados.



Figura - Vista adicionar produto

A vista de faturas é usada nos três submenus do menu “Faturas”, onde o utilizador escolhe se quer ver as faturas diárias, semanais ou mensais, com o objetivo de ir controlando o número de pedidos do dia, semana ou do mês.

Consoante o que o utilizador escolher, irá surgir na tabela do lado esquerdo todas as faturas correspondentes à escolha, mostrando apenas o número do pedido, a data e hora, e o valor total do pedido. O utilizador, ao selecionar uma linha e carregar no botão “Consultar”, terá acesso a todos os detalhes no lado direito, ou seja, a todos os detalhes do pedido: produtos, respetivas quantidades e valor numa tabela, número do funcionário que registou o pedido e contribuinte, se foi colocado.

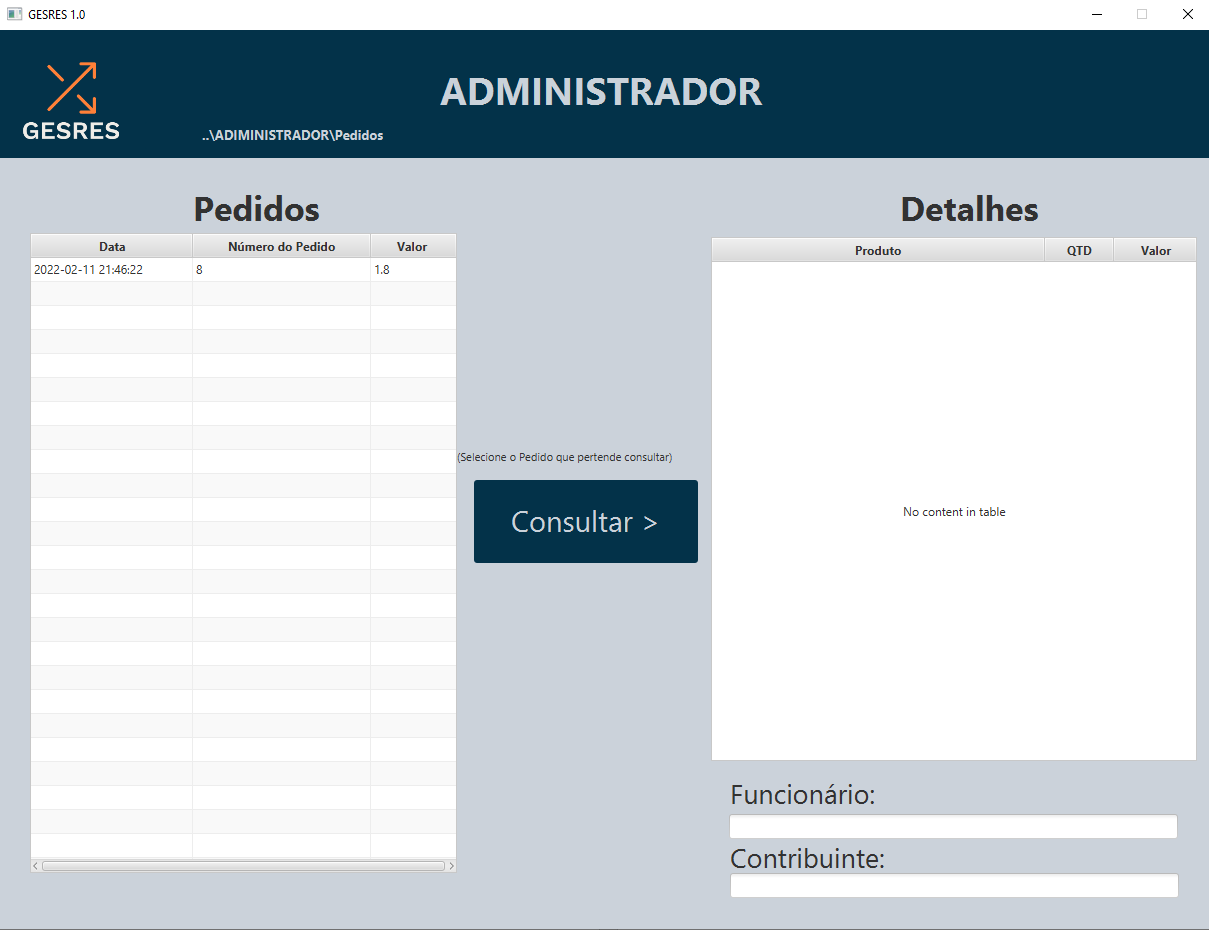


Figura - Vista faturas

Dashboard- Funcionário

A dashboard “funcionário” é composta por um total de sete vistas diferentes. A primeira vista da mesma é composta por dez botões numerados de um a dez, correspondendo ao número das mesas do estabelecimento. Esta vista foi elaborada com o intuito de controlar a disponibilidade das mesas do estabelecimento.

Ao carregar num dos botões, passa-se para a próxima vista, que mostra os detalhes da mesa em questão, o número correspondente à mesa. Quando a mesa está livre, o botão fica da cor verde, enquanto quando está ocupada, fica vermelho. No final existe um botão que serve para registar os pedidos de balcão, isto é, para quem não precisar de usufruir de mesa, este botão, quando selecionado, abre a vista para introduzir o código do funcionário.



Figura - Vista mesas

A vista de detalhes da mesa, criada para gerir cada mesa, é composta por uma tabela, para que, caso o pedido esteja ativo, mostra na mesma todo o pedido. A referida vista é composta quatro botões: “Novo Pedido”, que, quando selecionado, abre a vista introduzir o código de funcionário, “Editar Pedido”, que abre novamente a vista de realizar o pedido novo, mas com a função de acrescentar ao pedido antiga (uma forma encontrada para não realizar uma vista diferente), “Despachar Mesa”, que servirá para pôr o botão que corresponde ao número da mesa novamente verde, assinalando que a mesa está livre. Por último, existe ainda o botão “Pagamento”, onde é apresentada outra vista com o total do pedido escrito numa TextField, outra TextField para colocar o contribuinte do cliente caso queira, e uma ComboBox para selecionar o método de pagamento. Após carregar no botar “Finalizar”, o pedido é posto noutra tabela da base de dados “faturas” e a mesa fica automaticamente livre.



Figura - Vista detalhes mesas

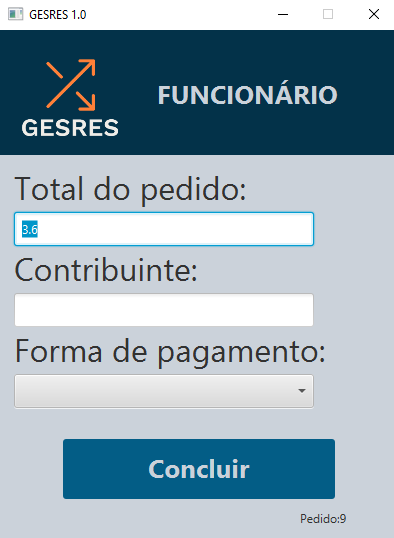


Figura - Vista pagamento

A vista de introduzir um código de utilizador é sempre mostrada antes de iniciar um novo pedido, de modo a ficar registado o funcionário que fez esse mesmo pedido. Esta é composta por uma TextField, que serve para introduzir um número, o qual, após carregar no botão concluir, é validado, verificando se existe o número na base de dados. Caso este não exista, é lançado um alerta referindo que o código é inexistente. Se existir o código, é aberta a vista para realizar o pedido com o nome do funcionário correspondente ao código.



Figura - Vista código funcionário

A vista de pedidos, criada de forma básica para que o funcionário possa registar um pedido de maneira simples e intuitiva, é composta por uma tabela onde o funcionário irá selecionar o produto que deseja continuar. Os produtos são mostrados na tabela conforme o tipo do produto desejado, clicando-se num dos seis botões, colocados acima da tabela (“Balcão”, “Sumos”, “Bebidas”, “Snacks”, “Doces”, “Cozinha”), e que vão buscar à base de dados todos os produtos correspondentes ao tipo pretendido. Para cada produto será obrigatório selecionar, na ComboBox, uma quantidade desejada, de 1 a 9, e, opcionalmente, uma observação ao produto. Também existe um campo, composto por uma TextField, para só indicar o valor se forem pedidos produtos a retalho.

Após a seleção dos itens obrigatórios, o funcionário terá que pressionar no botão “Adicionar”, que guardará os dados do produto num objeto da classe “Pedidos” e vai mostrando noutra tabela o pedido a ser formado.

Caso o funcionário tenha, por engano, cometido um erro em algum produto, terá a opção de remover o produto do pedido, selecionando o mesmo e pressionando no botão “Eliminar”, o qual irá remover o produto que havia sido gravado numa ObservableList.

Quando o pedido estiver concluído, o funcionário terá de pressionar no botão “Finalizar Pedido”, e este será guardado na base de dados.

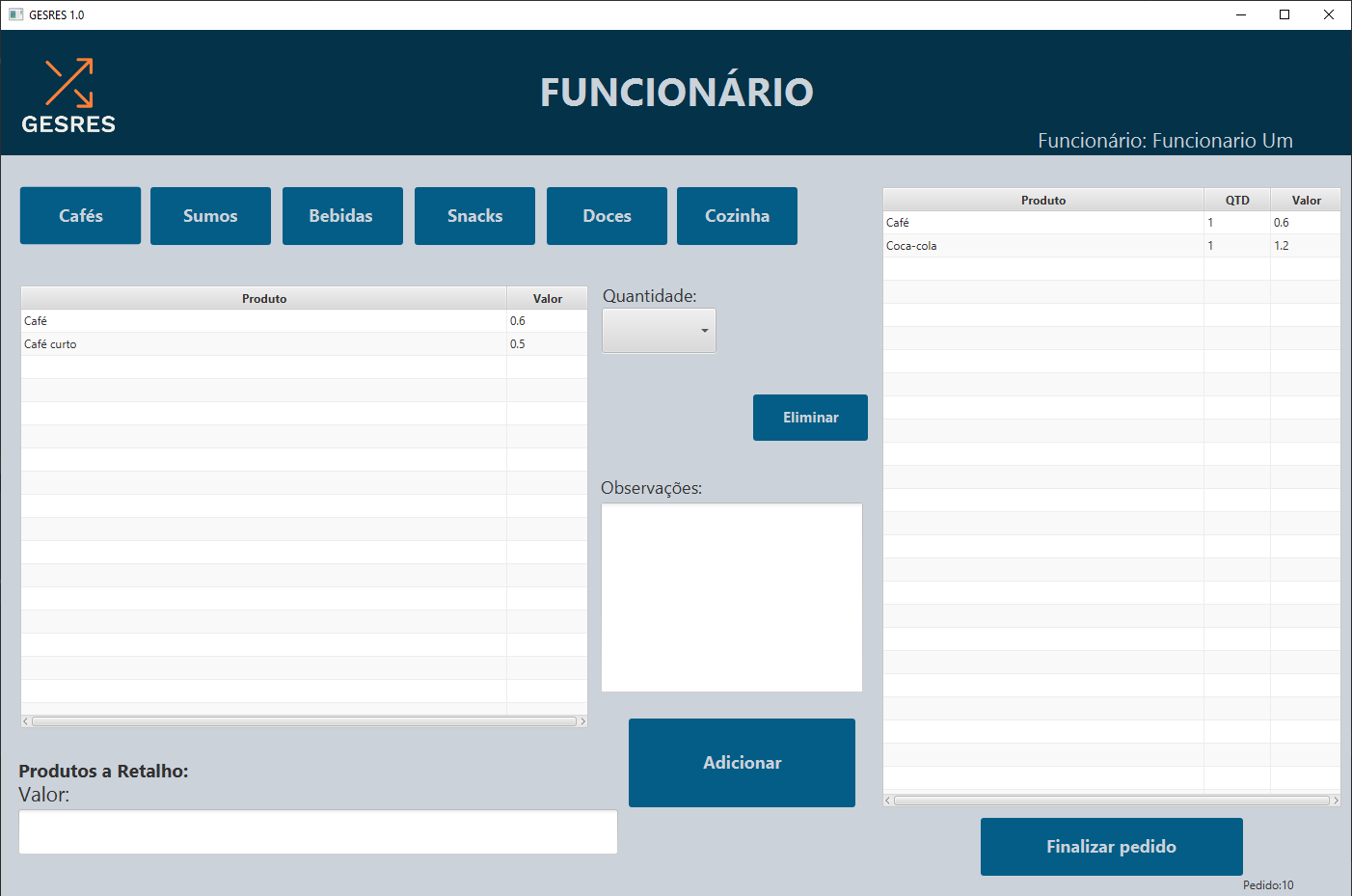


Figura - Vista pedidos

Dashboard- Cozinha

A dashboard da cozinha, primeiramente, era para ser realizada no programa principal, todavia, considerei mais adequado criar outro projeto à parte para a realização da mesma, com o intuito de enviar a parte do pedido que corresponde à cozinha.

Este projeto é composto apenas por duas vistas, *login* que foi aproveitado o do programa principal, contudo, neste, na ComboBox, só irá aparecer a opção de fazer *login* no utilizador “Cozinha”, e a outra vista dedicada aos funcionários da cozinha do estabelecimento.

A segunda vista é composta por uma ComboBox, sendo atualizada automaticamente de 30 em 30 segundos, que valida os pedidos ainda abertos, caso tenha algo para a cozinha. A mesma ComboBox será preenchida apenas com o número do(s) pedido(s). Após selecionar um dos pedidos da ComboBox, o funcionário terá que pressionar no botão “Ver Detalhes” e serão mostrados na tabela de detalhes o(s) prato(s) e as respetivas quantidades e observações, caso haja. Acima da tabela encontra-se uma TextField que mostrará o número do pedido em questão, o qual também passará para uma tabela - tabela de confeção - onde constará a lista de pratos a serem confecionados.

Para finalizar o pedido, o funcionário terá que selecionar na tabela de confeção o número de pedido que deseja finalizar, depois, pressiona no botão “Finalizar”.

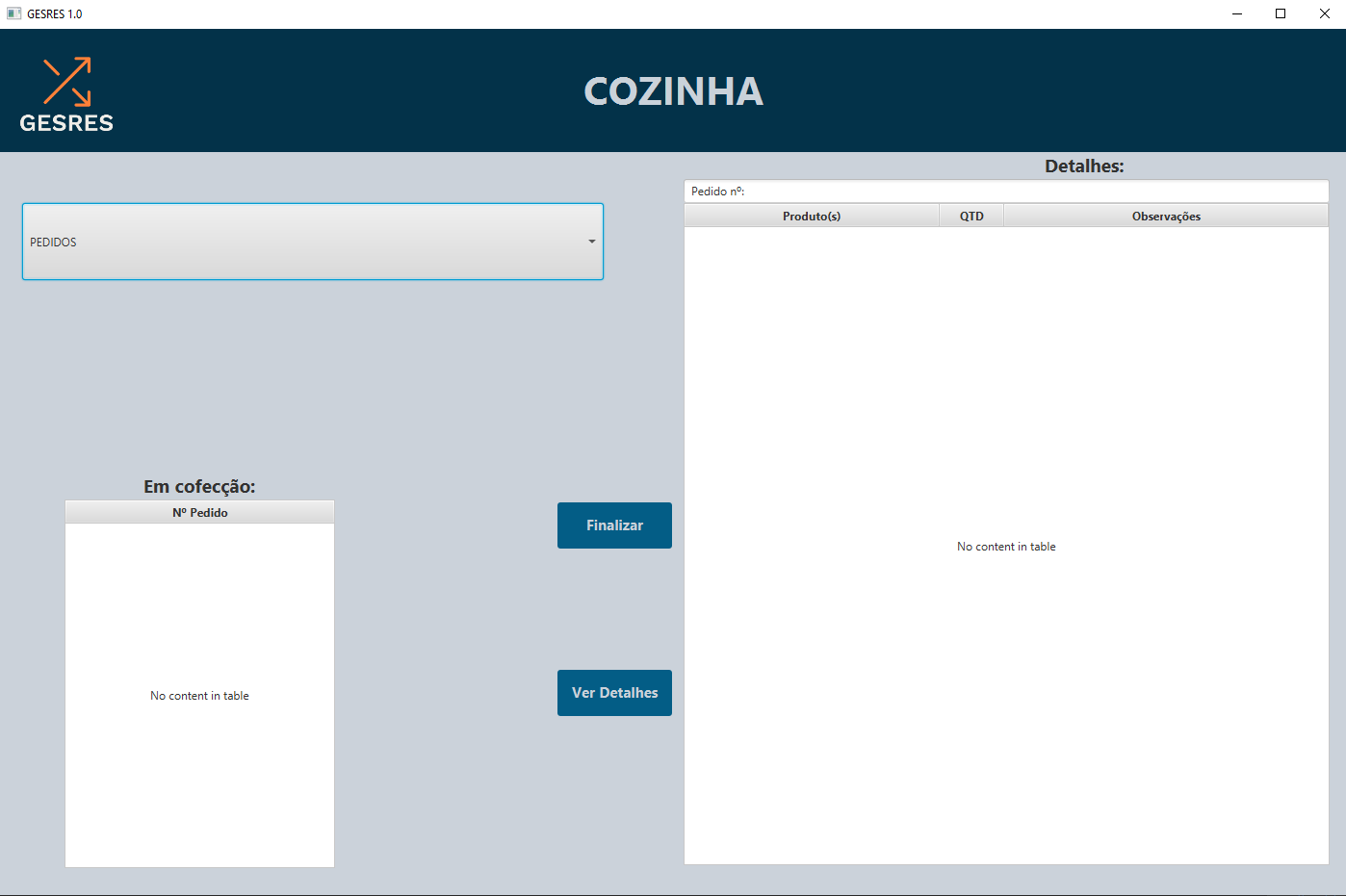


Figura - Vista cozinha

Organização do projeto

*Softwares*

Como *softwares* para a elaboração do mesmo, utilizei três *softwares* diferentes, nomeadamente ScenneBuilder, como já referido em cima, Intellij e MySQl Workbench.

No Intellij, usei a linguagem JAVA juntamente com bibliotecas da classe JAVAFX, para ligar e gerir os componentes FXML em JAVA, e a biblioteca “Mmysql-connector-java-8.0.25” para conectar o programa à base de dados. A linguagem foi essencialmente utilizada para a lógica do programa, como, por exemplo, para fazer a ligação à base de dados, gerir eventos do programa e efetuar cálculos.



Figura - Bibliotecas JAVAFX



Figura - Biblioteca MySQl-connector

Otimização do código

Sendo JAVA uma linguagem orientada a objetos, foram sendo criadas várias classes, com métodos que podem ser utilizados em diversos outros ficheiros, evitando, assim, a extensão e a repetição de código.

Por exemplo, na classe “MySQlConnection.java”, seriam declarados métodos de pesquisa na base de dados e, após importar a classe no ficheiro pretendido, seria possível usar qualquer método existente no ficheiro.

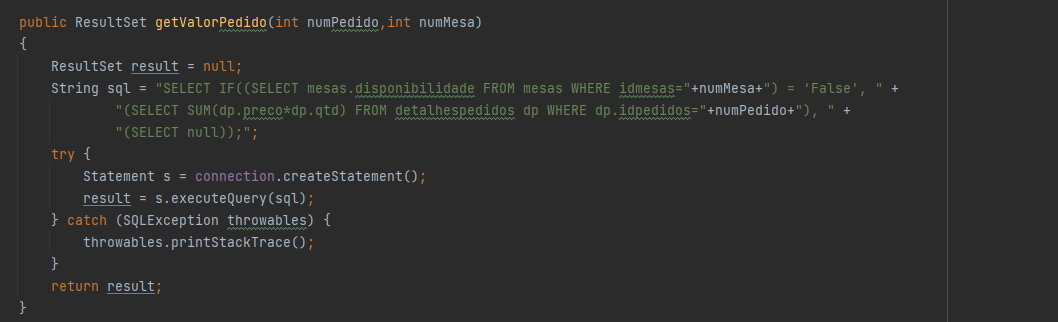


Figura - Exemplo função classe "MySQlConnection.java"



Figura - Import classe "MySQlConnection.java"

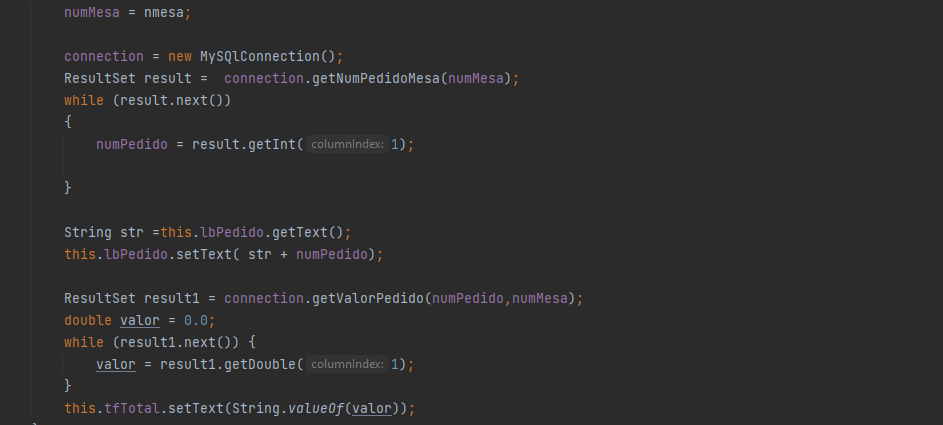


Figura - Retorno função da classe "MySQlConnection.java"

Organização dos ficheiros

Para a organização dos ficheiros do projeto, foi utilizado um método de arquitetura MVC (“Model-View-Controller”) utilizando-se as linguagens Java, com bibliotecas de JAVAFX, e FXML

MVC é um padrão de arquitetura de *sofware* responsável por contribuir para a otimização da velocidade de resposta aos pedidos do utilizador.

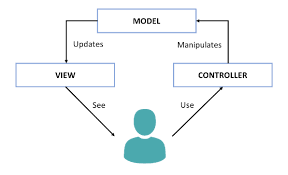


Figura - Arquitetura MVC

Na pasta “src” do meu projeto, criei as três pastas que constituem um projeto dessa arquitetura: “model”, “view” e “controller”. Dentro da pasta “model”, constam todas as classes criadas para um melhor tratamento dos dados. Na pasta “view”, estão todas as vistas do programa, ou seja, tudo aquilo a que o utilizador terá direito a ver na sua experiência. Por fim, a pasta “controller” contém todos os controladores correspondentes às vistas para gerir dados e ações das mesmas.

Para uma melhor organização, desta vez pessoal, na pasta “view” foi criada outra pasta,” adminViews”, onde existem as vistas correspondentes à dashboard de Admin que podem ser abertas no MenuBar da mesma. Assim, na pasta “controller” encontra-se a pasta “adminControllers” que tem como função controlar as vistas da pasta “adminViews”.

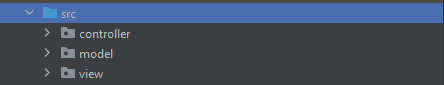


Figura - Pastas projeto MVC

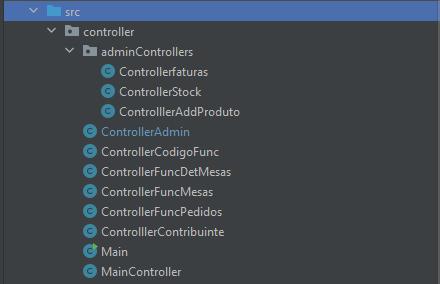


Figura - Pasta "controller"

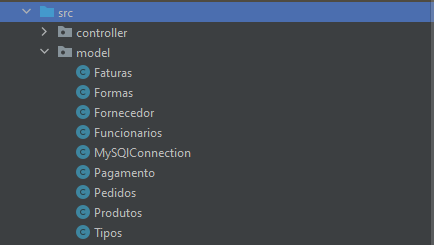


Figura - Pasta "model"

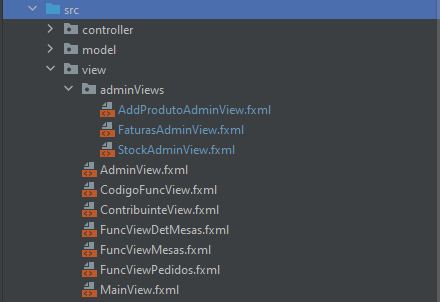


Figura - Pasta "view"

Base de dados

Para a base de dados, utilizei duas ferramentas: MySQl Workbench, para a criação e configuração da base de dados, e para alojar a mesma utilizei um servidor virtual alojado na AWS educate.

A razão de alojar a base de dados num servidor virtual, e não num servidor local, foi poder trabalhar com mais facilidade entre escola e casa, visto que cada alteração efetuada, seja em casa, seja na escola, ficava guardada no servidor e não na máquina em questão.

A base de dados foi ligada pelo ficheiro “dbConfig.properties”, por sua vez, colocado na raiz do programa, onde foram colocadas as credenciais de acesso à base de dados (“url”, “username”, “password”).

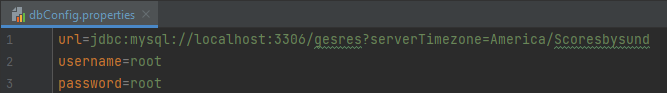


Figura - Ligação à base de dados

Estrutura base de dados

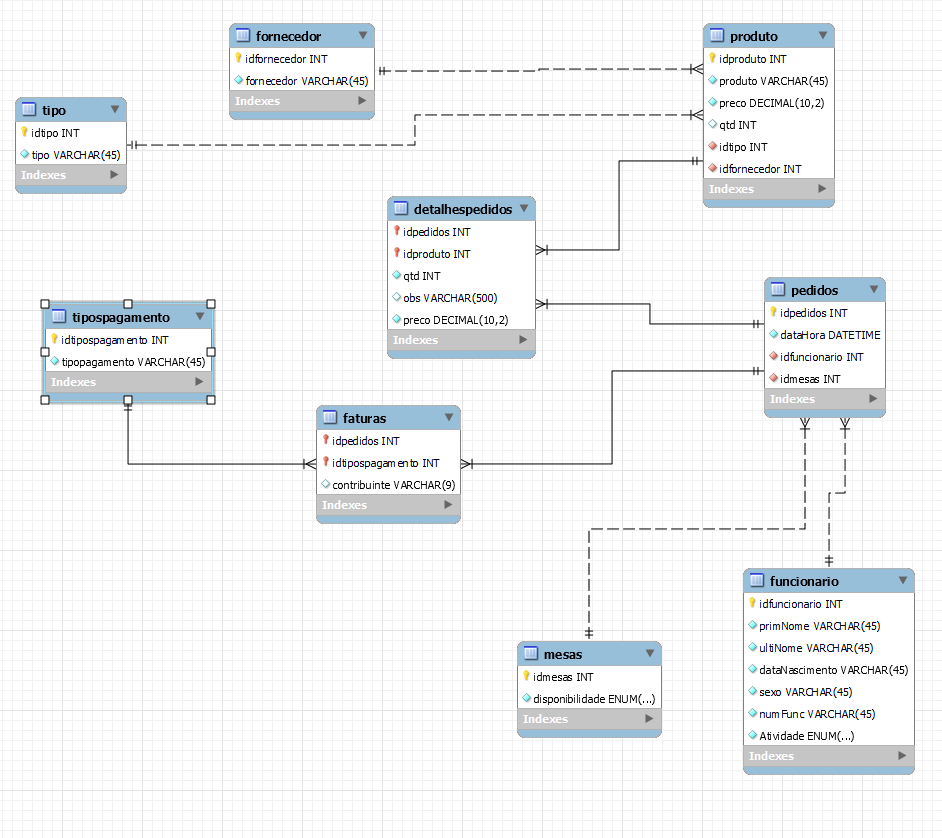


Figura - Modelo base de dados

A tabela “fornedor” é igualmente composta por duas colunas: “idfornecedor” e “fornecedor”, que não podem receber valores *null*. A primeira coluna “idfornecedor” é uma chave primária do tipo INT, já a segunda, “fornecedor”, é do tipo VARCHAR, podendo receber no máximo quarenta e cinco carateres.

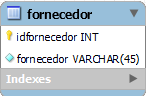


Figura - Tabela "fornecedor"

A tabela “tipo”, que foi criada para orientação de cada tipo dos produtos para venda, é composta por duas colunas: “idtipo” e “tipo”, que não podem receber valores *null*. À semelhança da tabela anteriror, a coluna “idtipo” é uma chave primária do tipo INT. Já a coluna “tipo”, sendo do tipo VARCHAR, pode receber no máximo quarenta e cinco carateres.

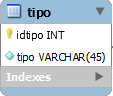


Figura - Tabela "tipo"

A tabela “produto” é composta por seis colunas: “idproduto”;” produto”;” preco”;” qtd”; “ idtipo” e” idfornecedor”. A coluna “idproduto” é uma chave primária do tipo INT e é auto incremento. A segunda coluna, “produto”, é do tipo VARCHAR, com um máximo de quarenta e cinco carateres, para receber o nome do produto. A coluna “preco” é do tipo DECIMAL(10,2) , ou seja, pode receber números com dez números inteiros e duas casas decimais. A coluna “qtd” é do tipo INT, recebendo a quantidade disponível em *stock* do produto em questão. As colunas “idtipo” e “idfornecedor” são chaves estrangeiras vindas das tabelas “tipo” e “fornecedor, respetivamente.

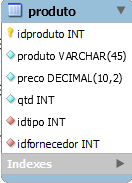


Figura - Tabela "produto"

A tabela “funcionarios” é composta por sete colunas: “idfuncionario”; “primNome”; “ultiNome”; “dataNascimento”; “sexo”; “numFunc” e “atividade”, nenhuma podendo ter valores *null*. A primeira coluna “idfuncionario” é uma chave primária do tipo INT e é auto incremento, ou seja, atribui automaticamente um id ao funcionário. As colunas “primNome”; “ultiNome”; “dataNascimento”; “sexo” e “numFunc” são do tipo VARCHAR, com um máximo de quarenta e cinco caracteres. A última coluna, “atividade”, está limitada a receber apenas dois valores:” Ausente” e “Presente”.

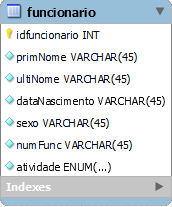


Figura - Tabela "funcionario"

Relativamente à tabela “mesas”, esta é composta por apenas duas colunas: “idmesas” e “disponibilidade”, não podedendo ter valores *null*. A primeira coluna, “idmesas”, é uma chave primária do tipo INT. A segunda coluna, “disponibilidade”, está limitada a receber apenas dois valores,” True” e “False”, ou seja, se estiver “True”, significa que a mesa está livre, caso contrário, estará “False”, assinalando que a mesma está ocupada.

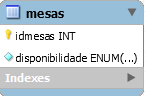


Figura - Tabela "mesas"

A tabela “pedidos” é composta por quatro colunas: “ idpedidos”, ”dataHora”, “idfuncionario”e “idmesas”, não podedendo ter valores *null.* A coluna “idpedidos” é uma chave primária do tipo INT e é auto incremento. A coluna “dataHora” é do tipo DATETIME, guardando a data, a hora, o minuto e o segundo do exato momento em que o pedido foi criado. As colunas “idfuncionario” e “idmesas” são de chaves estrangeiras vindas das tabelas “funcionário” e “mesas”, respetivamente.

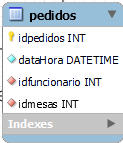


Figura - Tabela "pedidos"

A tabela “detalhespedidos” é composta por cinco colunas: “idpedidos”; “idproduto”; “qtd”;” obs” e “preco”. Nenhuma das colunas pode receber valores *null*, à exceção da coluna “qtd”, pois o produto pode não precisar de uma observação ou comentário. As duas primeiras colunas, “idpedidos” e “idproduto”, são chaves primárias estrangeiras, resultantes de duas ligações tipificadas das tabelas “pedidos” e “produto”, e formam uma chave primaria composta, ou seja, só pode existir uma combinação entre os dois IDs. A coluna “qtd” é uma do tipo INT, só aceitando números inteiros. A coluna “obs”, sendo do tipo VARCHAR, pode receber, no máximo, quinhentos caracteres, dado que pode haver um comentário extenso para o produto, apesar de esta coluna se aplicar mais aos produtos correspondentes à cozinha. A última coluna, “preco”, é do tipo DECIMAL(10,2), isto é, pode receber números com dez números inteiros e duas casas decimais.



Figura - Tabela "detalhespedidos"

A tabela “tipospagamento” é composta por apenas duas colunas: “idtipospagamento” e “tipopagamento”. A primeira coluna, “idtipospagamento”, é uma chave primária do tipo INT e é auto incremento. Já a segunda, “tipopagamento”, sendo do tipo VARCHAR, pode receber no máximo quarenta e cinco carateres.

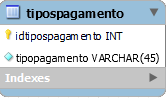


Figura - Tabela "tipospagamento"

Finalmente, a tabela “faturas” é composta por três colunas: “idpedidos”, “idtipospagamento” e “contribuinte”. As duas primeiras colunas, “idpedidos” e “idtipospagamento”, são chaves primárias estrangeiras, resultantes de duas ligações tipificadas das tabelas “pedidos” e “tipospagamento”; estas formam uma chave primária composta, ou seja, só pode existir uma combinação entre os dois Ids. A última coluna, “contribuinte”, pode receber valores *null* e está preparada para receber dados do tipo VARCHAR com o limite de nove carateres.

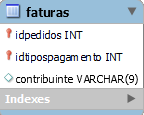


Figura - Tabela "faturas"

Conclusão

Bibliografia

Anexos