

Bruno Miguel Sousa

A21260212

Relatório de Programação Orientada a objetos

**Loja Guitarras p’Alugar**

Coimbra

2017

Introdução

Este trabalho foi desenvolvido com a ferramenta *Microsoft Visual Studio*, no qual o projeto em questão foi feito com a linguagem de programação c#.

Neste relatório irem constar os seguintes pontos:

* Estrutura do programa e apresentação de algumas partes de que possam ajudar nem compreensão do que é realizado nas funcionalidades programadas;
* Apresentação da estrutura de dados definidas nas classes (enum’s, list’s,…), acompanhado de uma breve explicação dessas mesmas escolhas;
* Justificações de opções tomadas durante o desenvolver do projeto;
* Pequena explicação para cada funcionalidade para um uso mais facilidade por parte do utilizador**.**

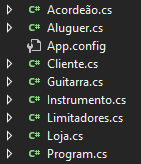
Requisitos Funcionais

O programa desenvolvido da ao utilizador os seguintes requisitos:

1. Cliente
   * Criar;
   * Alterar o nome;
   * Alterar o telefone;
   * Verificar alugueres que estejam em aberto;
   * Verificar histórico de alugueres
   * Ver dados de todos os clientes
   * Ver os x melhores clientes
2. Instrumentos
   * Criar
   * Verificar estado
   * Alterar preço diário do aluguer
   * Alterar valor do instrumento
   * Ver dados do instrumento
   * Ver histórico dos alugueres do instrumento
   * Ver dados de todos os instrumentos
   * Enviar instrumento para reparação
   * Receber instrumento reparado
3. Aluguer
   * Criar novo aluguer
   * Fechar aluguer
   * Ver alugueres que não estão ainda fechados
   * Ver alugueres que já foram fechados

Classes do programa

As classes criadas no desenvolver do projeto foram:

* ***Loja***: esta classe e a base de toda a informação e é esta que acede a todas as outras de forma a obter as informações necessárias

Nesta classe estão contidas a lista de clientes, guitarras, alugueres em aberto e alugueres já concluídos.

* ***Instrumento***: esta é uma classe abstrata que contem todas as funções e informações que são partilhadas por os instrumentos que podemos criar, todos eles partilham o ID que e *static* e de atribuição automática quando criado.
* ***Guitarra***: classe filha da classe Instrumento, esta é uma classe que nos permitira então criar instrumentos sendo estes de algum tipo que não instrumento.
* ***Acordeão***: classe filha da classe instrumento, no entanto esta classe não esta a ser utilizada no programa pois não existe uma opção para criar instrumentos deste tipo.
* ***Cliente***: nesta classe estão contidas todas as informações do cliente e funções que digam respeito ao mesmo, tal como mudança de nome e telefone.
* ***Limitadores***: nesta classe estão contidas algumas das limitações do programa, desta forma, podemos ter acesso as mesmas de forma rápida e fácil, sendo uma dela o máximo de alugueres que um cliente pode ter em simultâneo.
* ***Aluguer***: esta classe contem todas as informações de um aluguer e funções destinadas à verificação dos dados dos alugueres realizados, tando estas fechados ou não.

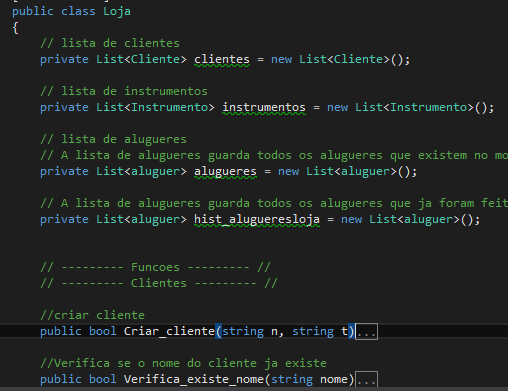
Formatação do código

O código no ***Program*** está organizado da seguinte forma:

* Variáveis para uso nos menus;
* Código do menu;
* Funções a serem chamadas por as opções do menu.

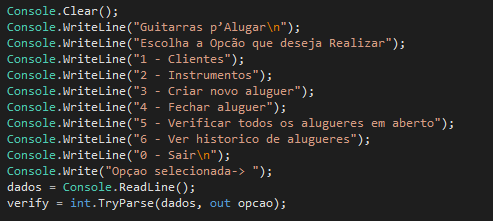
Esta organização foi assim construída por da uma facilidade e rapidez maior quando se quer aceder ao código das funções.

Já nas outras classes a organização seguida e semelhante:

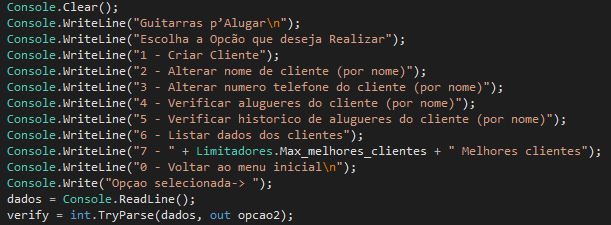
* Variáveis utilizadas para o armazenamento de informação;
* Funções para tratamento e modificação de dados

Menus

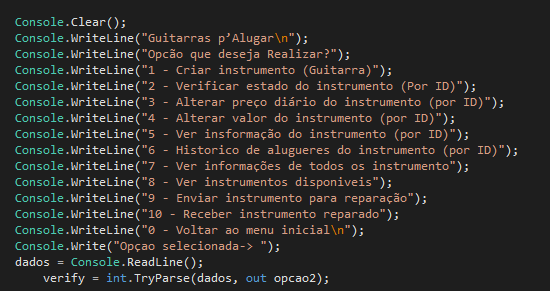
Neste programa contamos com três menus, todos eles programados por um *switch*, para que as opções sejam selecionadas de forma rápida e fácil para o utilizador e para o programa fazer a seleção e execução mais organizada.

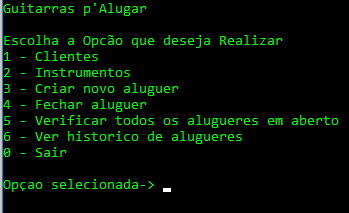
O menu principal que terá ligação para os submenus e que se destina a criação de alugueres, fecho de alugueres e outras informações que tenham haver com os alugueres.

O primeiro submenu nominado por Clientes tem as opções destinadas à manutenção de clientes, tal como criar ou alterar dados do cliente.



O segundo submenu com o nome Instrumentos conta com todas as funções que tenham haver com a criação de instrumentos na loja tal como a modificação ou acesso a informações de alugueres já terminados.

Todas estas opções são acedidas com o código que cada opção tem, este código e verificado e não deixa que o utilizador insira outros dados, no caso de isso acontecer o programa avisa o utilizador que não existe opção para os dados inseridos e volta a apresentar o menu.

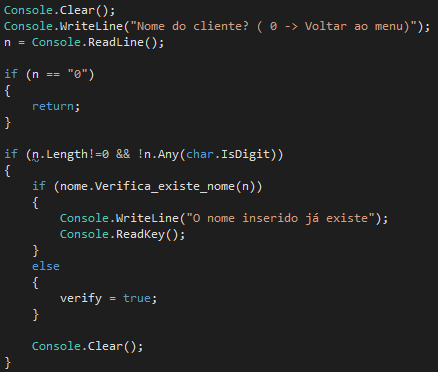
Aqui podemos ver os menus do utilizador e a mensagem em caso de a opção selecionada não ser valida.

Todas as opcoes do menu, quando selecionadas por engano, é dado ao utilizador a liberdade de inserir ‘0’ e assim voltar ao menu anterior sem quaquel problema.

Validação de dados

Quando os dados são inseridos pelo utilizador existem mecanismos para verificar se os dados são ou não validos para o que se quer fazer, por exemplo, temos a criação do cliente.

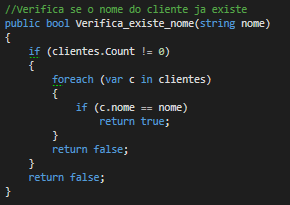
Na criação de um cliente, os dados inseridos têm de ser apenas alfanuméricos, este não pode existir no sistema ou ter algarismos, pois não será aceite caso isto não se cumpra.

Depois da inserção de um nome válido, o programa pede um numero de telefone, para que este seja aceite, o numero tem de ser constituído por 9 algarismos.

Esta verificação e feita sempre que o utilizador tem de inserir dados no programa, tanto para criação como para consulta de dados.

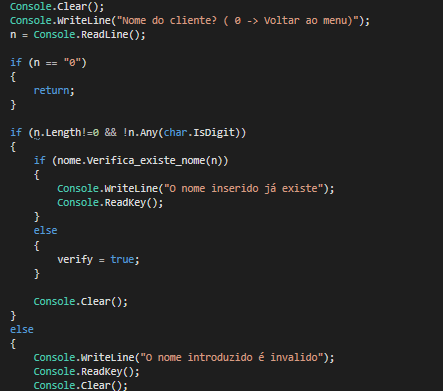
Funcionalidades do programa

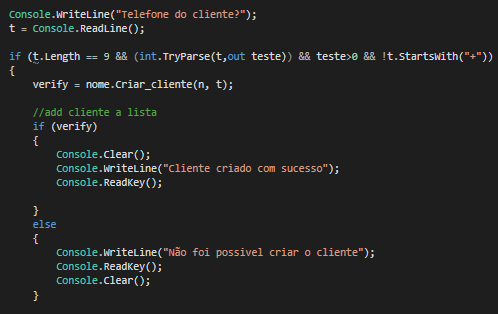
# Criar cliente

Quando o utilizador que criar um cliente, são pedidos o nome e o telefone para que seja então criado o cliente.

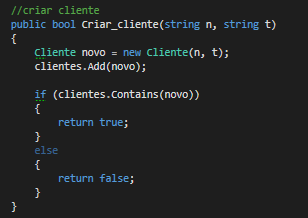
Estes dados são validados e apenas e criado o cliente caso:

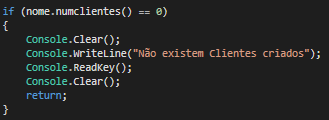
O nome dado seja constituído por caracteres alfanumérico e não exista um cliente com o nome igual;

O numero contenha 9 algarismos. Para um melhor funcionamento desta função, poderíamos também meter uma verificação no numero, para que não haja números iguais em clientes diferentes.



Caso ambos os dados sejam validos, criamos o cliente na função dentro da loja.

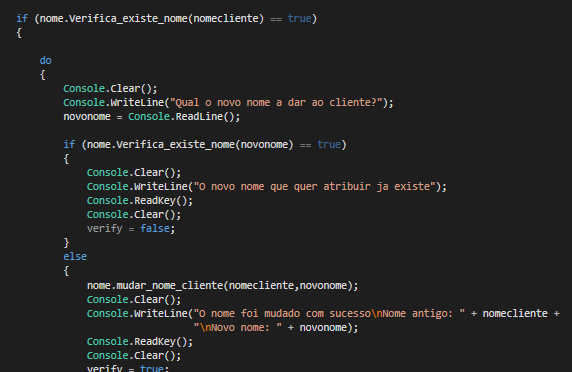


Caso o programa não contenha clientes criados, as opções que precisem de clientes não poderão ser executados e iram mostrar uma mensagem a dizer que não existem clientes criados.

# Alterar nome (Cliente)

Função que permite alterar nome do um cliente, esta função pede ao utilizador o nome que esta guardado, procura na lista e depois pede para que se de o novo nome.

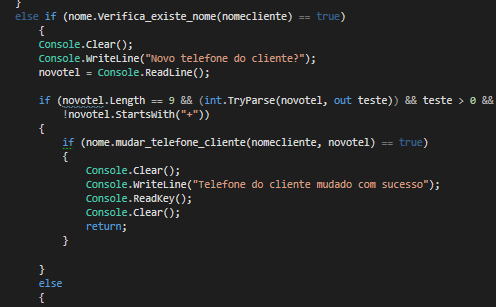
É feita a verificação do novo nome e se não existe nenhum cliente com o novo nome que se quer dar ao cliente.

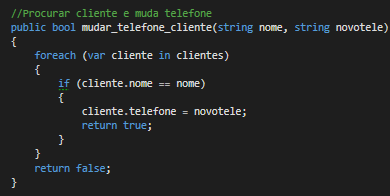


Quando esta mudança e feita, o programa mostra ao utilizador uma mensagem a dizer que a alteração foi efetuada com sucesso, o nome antigo e o novo nome atribuído.

# Alterar telefone (Cliente)

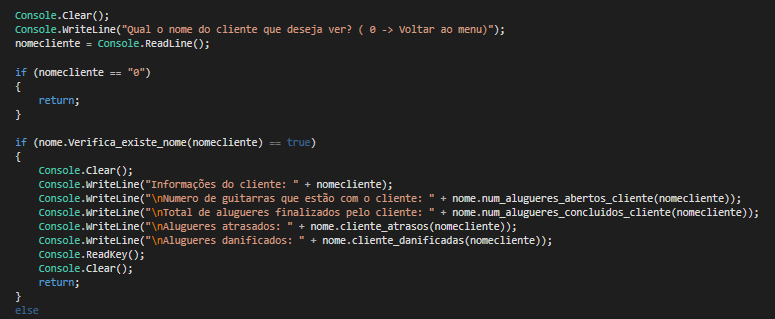
Esta função pede ao utilizador o nome do cliente ao qual se quer alterar o numero de telefone.

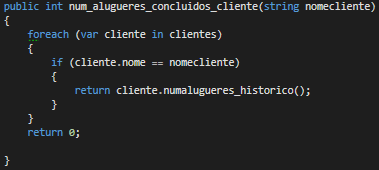
Depois de verificar se o cliente com o nome dado existe, este pede então o novo numero que se quer dar, antes de atribuir o novo numero, este passa por as verificações tal como quando criamos o cliente, para ter a certeza que é valido.

Quando esta alteração e feita com sucesso, e mostrada uma mensagem ao utilizador para que saiba que este foi alterado.

# Estado (Cliente)

Esta Função mostra no ecrã as informações do cliente, dado o nome por o utilizador.

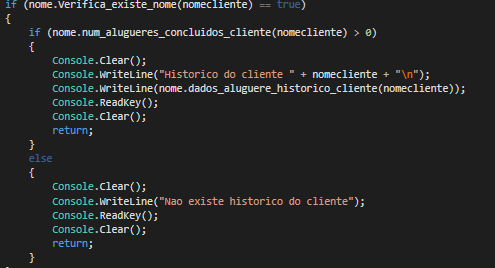
Todas as informações são pedidas por funções ‘GET’ à loja e imprimidas no ecrã.

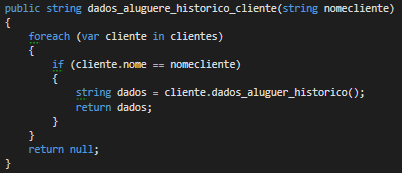


A informação dada por esta função é:

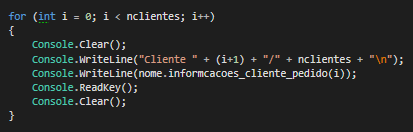
* Nome do cliente;
* Numero de guitarras alugadas no momento;
* Total de alugueres que já foram finalizados;
* Alugueres finalizados que foram entregues atrasados;
* Alugueres finalizados que foram entregues com o instrumento danificado.

# Histórico alugueres (Cliente)

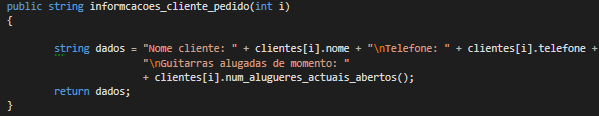
Esta função, pede ao utilizador, o nome do cliente do qual se quer ver o histórico de alugueres, isto é, os alugueres que o cliente fez, mas que já terminaram com a entrega dos instrumentos.

Quando o nome e valido, a função verifica se o cliente tem ou não alugueres no seu histórico, no caso de ter, são mostrados, caso não tenha é então mostrada uma mensagem a dizer que o cliente não contem histórico de alugueres.

# Listar dados (Todos clientes)

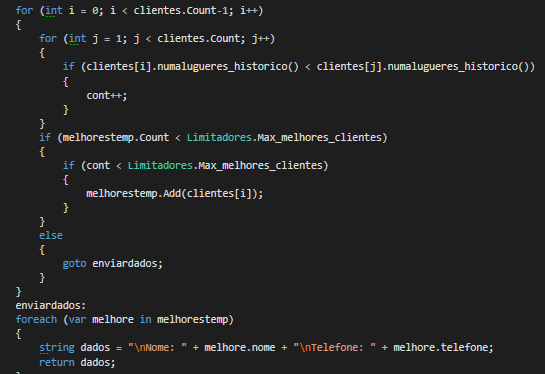
Esta função que mostra os dados de todos os clientes, a partir de um ciclo *for.*

A função corre o numero de vezes igual ao numero de clientes que a lista de clientes tenha, para que assim mostra a informação de todos, sem que salte nenhum.



# Melhores clientes

Esta função mostra os melhores clientes, ou seja, os que mais alugueres concluídos tenham.

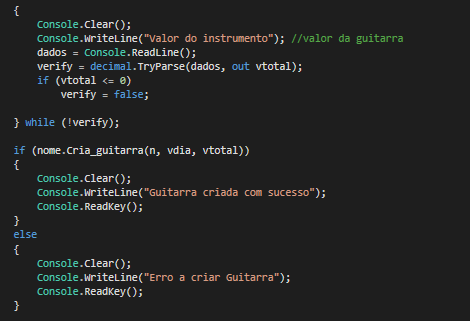
Para isso são usados 2 ciclos *for* para que se possa comprar todos os clientes entre si e se saiba de todos quais os melhores.

Usamos também aqui o limitador para verificar se o utilizador e o melhor ou não, isto é, se não houver mais de 3 clientes com mais alugueres do que o cliente que estamos a comparar, então este será copiado para uma lista temporária, da qual depois vamos buscar os dados dos clientes que nela estiverem.

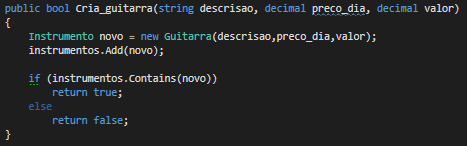
Este limitadore pode ser alterado rapidamente na classe *Limitadores, caso queiramos mais ou menos clientes na lista.*

# Criar instrumento (Guitarra)

A função de criar instrumentos esta definida para criar apenas guitarras, pois a criacao de acordeões esta definida como *private* para que não seja criado nenhum acordeão.

Para a criação de acordeões, apenas teremos de copiar a função das guitarras e criar objetos da class Acordeão.

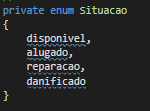
Na criação de um objeto guitarra, é pedido ao utilizador uma pequena descrição do instrumento, o preço a pagar por dia de aluguer e o valor do instrumento, todos estes campos fazem verificação dos dados introduzidos.

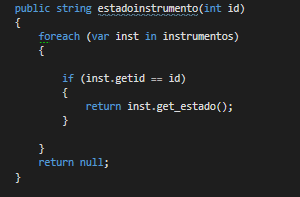


Para uma melhoria nesta função, seria não deixar que o preço de aluguer seja maior que o valor do instrumento e o valor do instrumento não poder ser menor do que o preço de aluguer por dia.

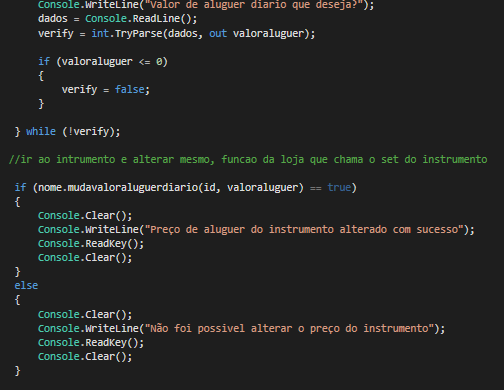
# Verificar o estado (Instrumento)

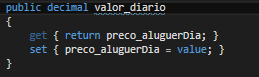
Esta função pede ao utilizador o ID do instrumento e mostra o seu estado.

O estado do instrumento esta definido com uma *enum* e apenas pode ter os valores que lá estão.

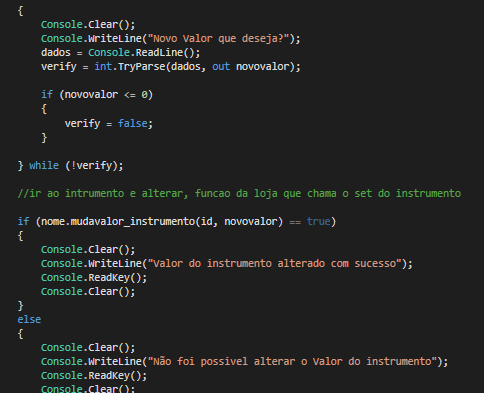


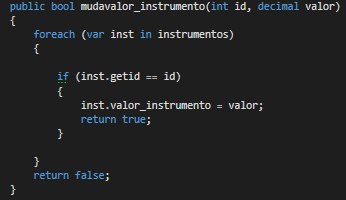
# Alterar o preço diário de aluguer (Instrumento)

Esta função pede ao utilizador o ID do instrumento ao qual se quer mudar o valor a pagar por dia de aluguer.

Uma melhoria possível a fazer nesta função seria não autorizar a mudar o valor de aluguer, caso o instrumento a que se quer alterar o preço estiver com a sua situação como alugado.

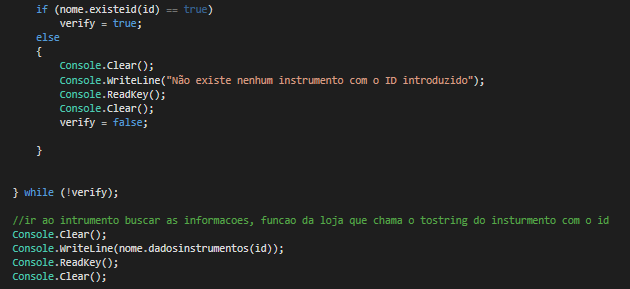
# Alterar o valor (Instrumento)

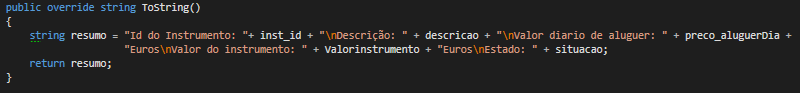
Esta função pede ao utilizador o ID do instrumento ao qual se pretende mudar o valor.

Tal como a mudança do preço de aluguer, seria bom não deixar que este valor fosse menor que o preço do aluguer diário do equipamento.

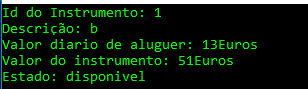
Outro ponto que também poderia fazer sentido era deixar apenas mudar o valor do instrumento e o valor de aluguer caso este fosse reparado.

# Ver informações (Instrumento)

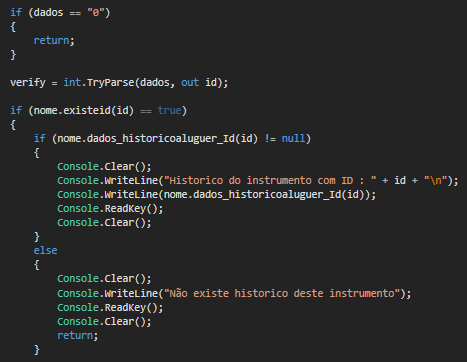
Esta função pede ao utilizador o ID do instrumento do qual se quer ver a informação.

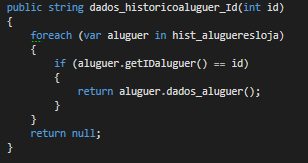
Esta função chama outra criada na *Loja* recebe o id, procura o instrumento na *lista*, com o *id* passado e depois chama a função *Tostring* criada no *Instrumento,* que retorna uma *string* com os dados já guardados.

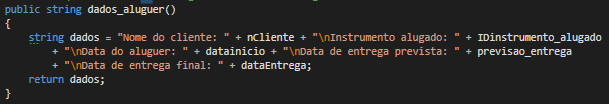
A informação apresentada será:

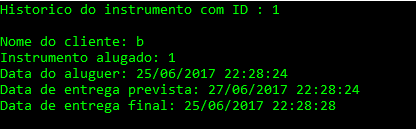
* ID;
* Descrição;
* Valor diário a pagar por aluguer;
* Valor do instrumento;
* Estado.

# Histórico de alugueres (Instrumento)

Esta função mostra o histórico do instrumento, este escolhido por o utilizador dando o ID.

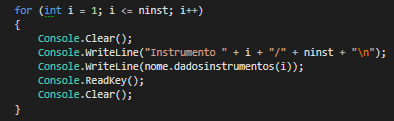


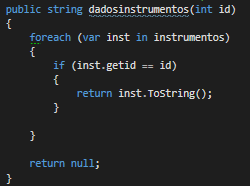
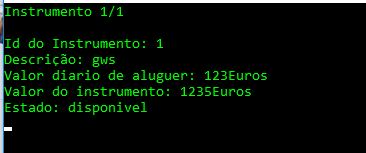
A função *dados\_historicoaluguer\_Id* vai retornar depois os dados recebidos por a função chamada.

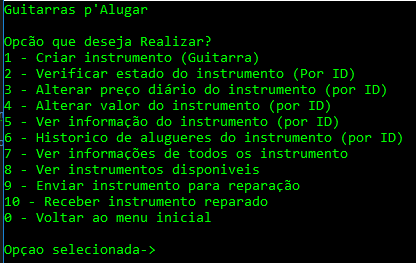
Os dados apresentados são todos os alugueres em que o instrumento alugado, tem o ID que o utilizador deu quando pedido um ID.

Esta opção apenas mostra alugueres que já tenham sido terminados, cado o instrumento esteja alugado, este não será mostrado, pois ainda não foi concluído.

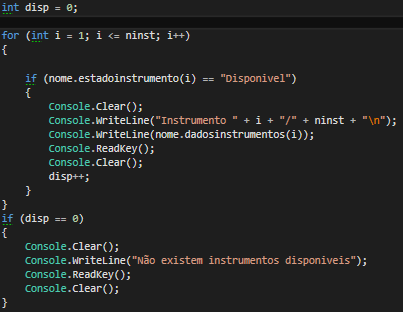
# Informação (Todos os instrumentos)

Esta função serve para correr a lista dos instrumentos e mostrar ao utilizador a informação básica dos instrumentos que existem, não interferindo a sua situação no momento.

Programada com um *for* para que assim se percorra todos os instrumentos sem saltamos nenhum.

 A informação dos instrumentos pode ser passada premindo qualquer tecla, no caso de não haver mais instrumentos, voltamos ao menu dos instrumentos.

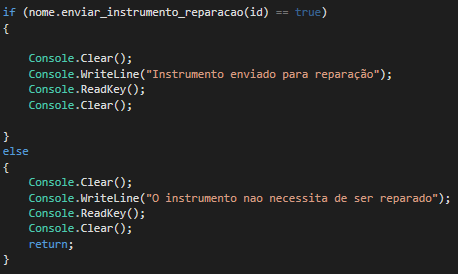
# Lista disponíveis (Todos Instrumentos)

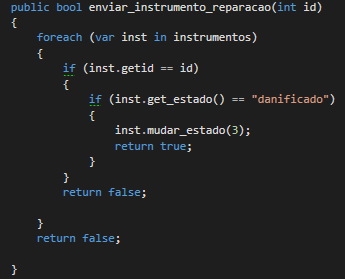
 Esta função mostra ao cliente todos os instrumentos que a sua situação, quando e chamada a função, seja ‘disponível’, qualquer instrumento que tenha um estado diferente não é mostrado ao cliente.

A função e constituída por um ciclo *for* com um *if* dentro, que caso o instrumento tenha o estado como disponível, mostra a informação desse mesmo instrumento.

Usando a variável *disp* podemos saber se no fim do ciclo *for* foi ou não mostrado algum instrumento ao utilizador, caso não tenho sido mostrado nenhum instrumento é mostrada a mensagem:

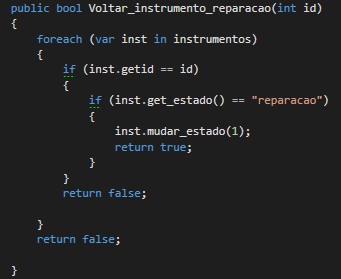
# Enviar para reparação

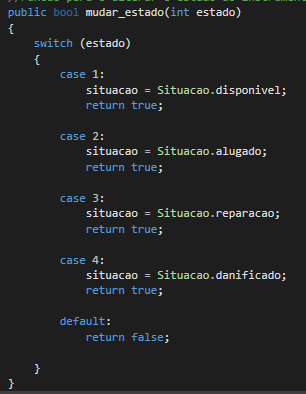
Esta função pede ao utilizador o ID do instrumento que este deseja mandar para reparação, antes de mandar para reparação o instrumento, a função vai verificar se o estado deste é ‘danificado’, caso não seja a função diz que o equipamento não precisa de reparação.

Caso o instrumento tenha o estado como ‘danificado’ então a função muda o estado, do instrumento dado pelo o utilizador, para ‘reparação’.

Este método assim previne que instrumentos que não precisam de ser reparados fiquem sem poder ser alugados.

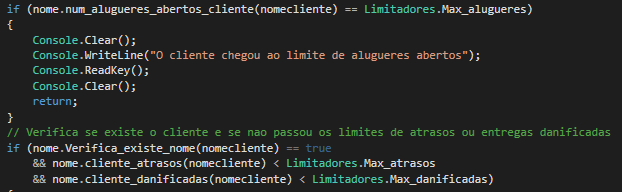
# Receber da reparação

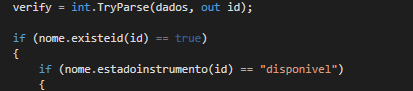
Esta função faz quase a mesma coisa que a ‘enviar para reparação’, pede o ID do instrumento ao utilizador e verifica se instrumento em questão estava mesmo em reparação e muda o seu estado para ‘disponível’.

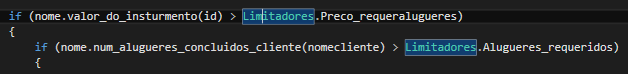
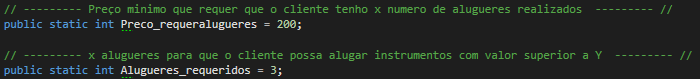
A mudança de estado e executada com um método que recebe um valor entre 1 e 4, em que cada um equivale a uma opção no *switch*, alterando assim o estado do instrumento.

# Criar aluguer

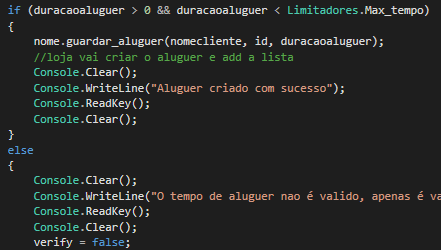
Esta função serve para a criação de alugueres, começa por verificar se existem clientes e instrumentos criados, pois sem ambos ou um dos dois, não podemos registar um aluguer.

Posto isto, pede o nome de um cliente e verifica se este existe, não esta no limite de alugueres e se não passou os limitadores de atrasos e instrumentos entregues danificados.

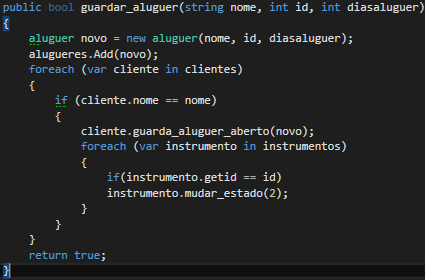
 Caso esteja tudo bem, pede o ID de um instrumento e faz também a verificação se este existe e se esta disponível para alugar.

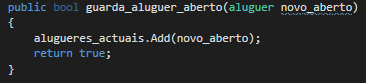
Depois de verificar que esta disponível o instrumento, ainda temos de verificar o valor do instrumento, caso o seu valor seja maior que o limitador, teremos de verificar o limitador de alugueres que e preciso para alugar esse instrumento.

Caso o valor não passo o limitador, não é feita a verificação de quantos alugueres tem o cliente.

Com todos estas verificações passadas, o programa pede então o numero de dias que o cliente quer alugar o instrumento, aqui entra outro limitador que limita o máximo de dias que um instrumento pode ser alugado.

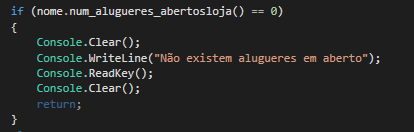
Quando o tempo de aluguer valido, passamos a criar o aluguer.

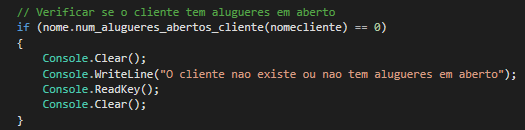
A função recebe então todos os dados que o utilizador inseriu e que foram validos.

Esta cria então um objeto do tipo aluguer com os dados e adiciona a lista de alugueres da loja, depois procura o cliente que fez o aluguer e vai guardar também este objeto na lista de alugueres abertos do cliente.

# Fechar aluguer

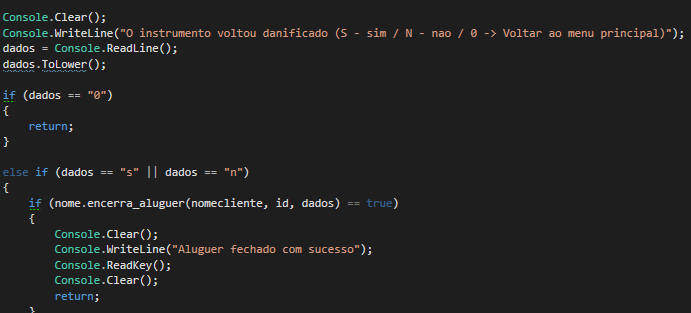
Esta função permite ao utilizador finalizar um aluguer.

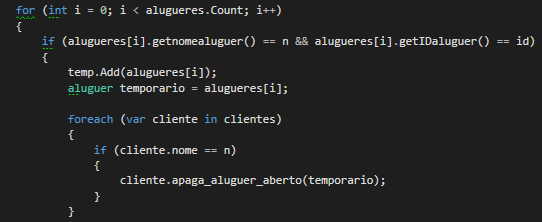
Ao correr a função, esta verifica se a lista de alugueres em aberto esta vazia ou não, pois caso esta esteja fazia, não há alugueres a finalizar.

Caso haja alugueres, pede o nome do cliente e verifica se este existe e se tem alugueres em aberto.

Se o cliente tiver alugueres em aberto, irá pedir o ID do instrumento para verificar se existe então alguém aluguer do cliente, em que o instrumento tenho esse ID.

Se verificar que existe o tal aluguer, pergunta ao utilizador se o instrumento vem danificado ou não, pedindo um ‘s’ para sim e um ‘n’ para não.

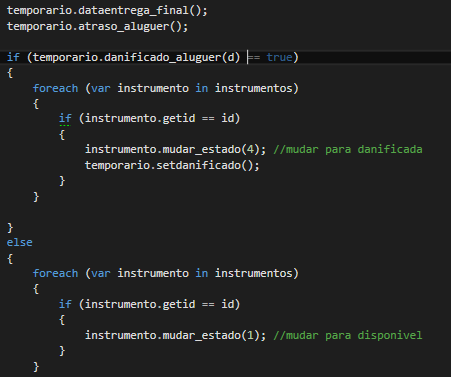


Então chama a função para encerrar o aluguer, com os dados recebidos.

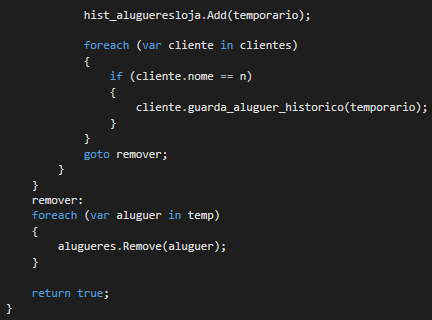
Esta função vai então aos alugueres em aberto e guarda o aluguer em questão numa variável temporária, e numa lista temporária o aluguer sem alterar nada.

Depois vai a lista dos clientes e apaga da lista de alugueres do cliente o aluguer igual ao temporário.

Depois de apagar o aluguer da lista do cliente, vai adicionar a data de entrega ao objeto temporário, verificar se houve atraso na entrega e ver se o equipamento veio danificado.

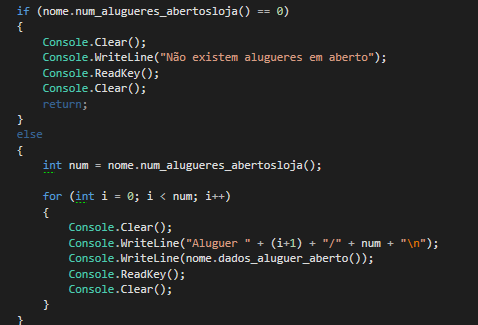
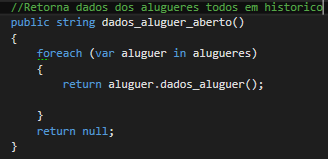
No caso de este vir danificado, o estado e alterado para ‘danificado’, se não, e alterado para ‘disponível’.

É guardado no temporário a forma de como foi entregue o instrumento, para que seja depois possível consultar.

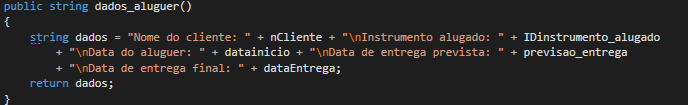
Depois de todos os dados preenchidos o aluguer temporário e guardado na lista de alugueres temporários da loja e do cliente que o fez.

No fim de tudo vai a lista temporária que tem o aluguer tal como esta na lista de alugueres abertos e vai procurar por ele e apagado, de forma a que apenas fiquei o aluguer com todos os dados preenchidos na lista de histórico de alugueres.

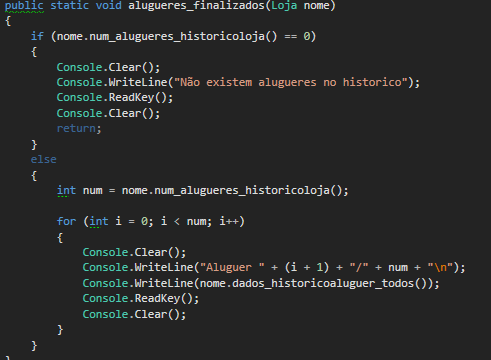
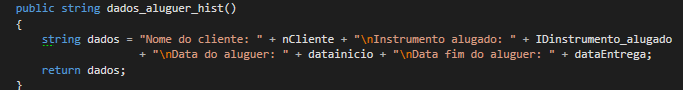
# Ver alugueres em aberto

Esta função vai buscar a lista de alugueres em aberto da loja e mostra as suas informações.

A função vai então retornar uma *string* com os dados referente ao aluguer.



# Ver lista de alugueres já fechados

Esta função faz a mesma coisa que o método acima referido, só que ao invés de ser os alugueres em aberto, irá correr a lista de histórico de alugueres, ou seja, ou alugueres concluídos.

# Conclusão

Apesar da falta de funcionalidades de podem melhorar bastante o funcionamento do programa, como o caso de guardas os dados em ficheiros para que quando o programa e fechado os dados não sejam perdidos ou a função que era pedida de criar um ficheiro com os alugueres que estão em atraso.

Contudo ao desenvolver este programa deparei-me com vários pormenores que podem sempre fazer com que os programas desenvolvidos podem sempre ser melhorados.