

MANUAL SISTEMA PARA HOTEL

GRUPO CLIENTES

EDUARDO AFONSO SOBRAL SANTOS TAVARES
CARLOS EDUARDO PEREIRA SILVA
FELIPE MENDONÇA DO SACRAMENTO
JOSÉ GEISON DOS SANTOS
SAMUEL SILVA DOS ANJOS

Introdução:

Este documento tem como finalidade ajudar na compreensão da parte do software desenvolvido pela equipe clientes, explicando suas funcionalidades e detalhes de forma mais clara, com o uso de imagens e trechos de códigos feito pela equipe. Essa parte do todo, tem por finalidade fazer o papel inicial e o final, quando um cliente chega em um hotel e solicita um quarto, que é o cadastro e suas diretrizes e o tratamento do fim da estadia.

Descrição Narrativa :

O grupo cliente ficou com a parte de clientes, nessa parte do software, o código faz o cadastro de clientes que tem interesse em alugar um quarto, logo depois temos outras opções como: listar clientes do arquivo, editar informações e exclusão de um cliente que já tenha terminado a sua estadia no hotel.

Falando um pouco mais, dessa parte do projeto do grupo que ficou com a parte de clientes, é fundamental em primeiro quesito cadastrar clientes em um arquivo, para que depois pudéssemos ter acesso à eles. Com os usuários em um arquivo, podemos listar a quantidade e os dados de cada cliente e, assim se necessário temos como fazer mudanças nos dados do clientes e logo depois regravar esses dados específicos de volta no arquivo e, por fim, como já foi citado acima temos a função excluir que retira os dados do respectivo cliente do arquivo, assim gerando um novo arquivo, só com os usuários que ainda permanecem no hotel.

Descrição do Caso de Uso:

Para operar as funcionalidades Cliente, vamos utilizar de imagens para auxiliar na explicação do caso de uso.

- Primeiro nós temos a tela inicial das funcionalidades que a equipe clientes criou:



Aqui, temos as opções de navegação pelo programa: Cadastrar cliente, listar cliente, modificar cliente e excluir cliente.

- Seguindo em frente, o segundo passo é nos cadastrarmos um cliente, onde é pedido uma série de dados pessoais:

NOME:	Samuel		
CPF:	12345678909	NACIONALIDADE:	Brasileiro
PROFISSÃO:	Estudante		
ENDEREÇO:	Itabiana		
IDADE:	23	SEXO [M F O]:	M

Depois de cadastrar o usuário do hotel aparecerá uma mensagem de confirmação indicando que o cliente foi cadastrado. Imagem logo a seguir:



- Em outro módulo, se quisermos modificar um dado de um cliente podemos editá-lo, é só entrar no módulo Modificar Cliente que você terá a opção de editar qualquer informação de um usuário qualquer :

Two screenshots of a user registration form. The top screenshot shows a form with fields for NOME, CPF, NACIONALIDADE, PROFISSAO, ENDERECO, IDADE, and SEXO. The bottom screenshot shows the same form with the name 'Samuel' and address 'Aracaju' entered.

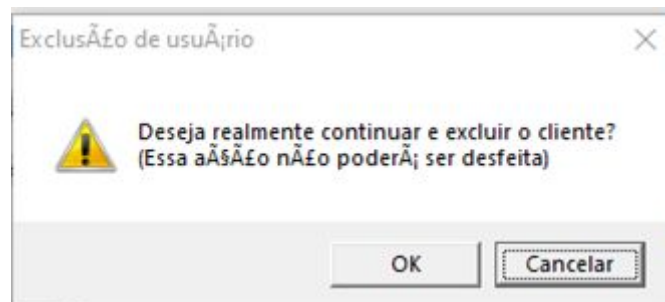
NOME:	NULL		
CPF:	12345678909	NACIONALIDADE:	Brasileiro
PROFISSAO:	Estudante		
ENDERECO:	Itabaiana		
IDADE:	23	SEXO [M F O]:	M

NOME:	Samuel		
CPF:	12345678909	NACIONALIDADE:	Brasileiro
PROFISSAO:	Estudante		
ENDERECO:	Aracaju		
IDADE:	23	SEXO [M F O]:	M

- E por fim temos a função excluir, que quando um cliente terminar sua estadia ele será listado fora do arquivo pelo manuseador do sistema, onde será pedido as informação do hóspede que se deseja excluir. E logo depois aparecerá uma mensagem pedindo se quer mesmo excluir o cliente. Imagem abaixo:

NOME:	NULL		
CPF:	12345678909	NACIONALIDADE:	Brasileiro
PROFISSAO:	Estudante		
ENDERECO:	Aracaju		
IDADE:	23	SEXO [M F O]:	M

A seguir a mensagem se realmente deseja excluir.



Alguns trechos de códigos :

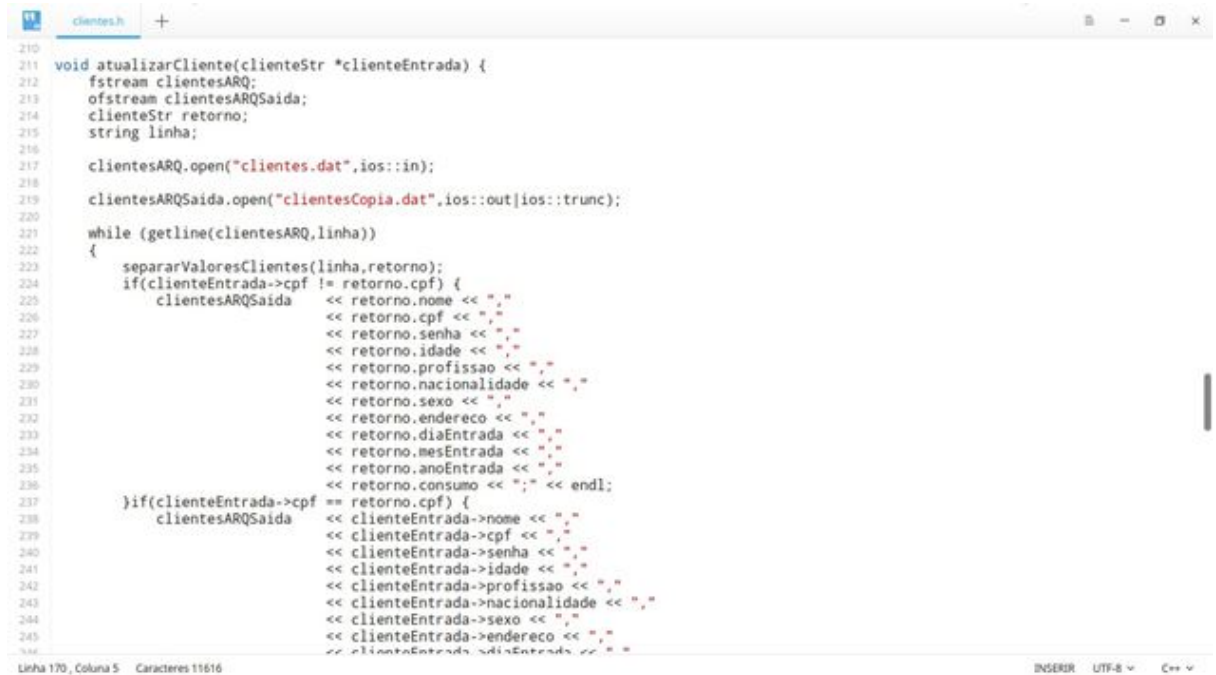
Cadastre um usuário e será chamado a biblioteca cpf.h para fazer a verificação se cpf é válido.

```

155 }
156
157 void cadastrarCliente(cliente *cliente) { //armazena no final do arquivo o usuário recebido como ponteiro
158     fstream clientesARQ;
159     time_t rawtime;
160     struct tm *info;
161
162     time(&rawtime);
163     info = localtime(&rawtime);
164
165     cliente->diaEntrada = info->tm_mday;
166     cliente->mesEntrada = info->tm_mon+1;
167     cliente->anoEntrada = 1900+info->tm_year;
168
169     clientesARQ.open("clientes.dat",ios::app);
170
171     clientesARQ << cliente->nome << ","
172     << cliente->cpf << ","
173     << cliente->senha << ","
174     << cliente->idade << ","
175     << cliente->profissao << ","
176     << cliente->nacionalidade << ","
177     << cliente->sexo << ","
178     << cliente->endereco << ","
179     << cliente->diaEntrada << ","
180     << cliente->mesEntrada << ","
181     << cliente->anoEntrada << ","
182     << cliente->consumo << ";" << endl;
183
184     clientesARQ.close();
185 }
186
187 cliente* buscarCliente(string cpf) { //procura o usuário que tem o equivalente CPF e retorna toda a struct pra usar os dados no
188     programa
189
190     fstream clientesARQ;

```

Para atualizar dados dos clientes cadastrados é chamada essa função que irá mover os dados de um arquivo para outro, assim atualizados.



```
210
211 void atualizarCliente(clienteStr *clienteEntrada) {
212     fstream clientesARQ;
213     ofstream clientesARQSaida;
214     clienteStr retorno;
215     string linha;
216
217     clientesARQ.open("clientes.dat", ios::in);
218
219     clientesARQSaida.open("clientesCopia.dat", ios::out | ios::trunc);
220
221     while (getline(clientesARQ, linha))
222     {
223         separarValoresClientes(linha, retorno);
224         if(clienteEntrada->cpf != retorno.cpf) {
225             clientesARQSaida << retorno.nome << " "
226                               << retorno.cpf << " "
227                               << retorno.senha << " "
228                               << retorno.idade << " "
229                               << retorno.profissao << " "
230                               << retorno.nacionalidade << " "
231                               << retorno.sexo << " "
232                               << retorno.endereco << " "
233                               << retorno.diaEntrada << " "
234                               << retorno.mesEntrada << " "
235                               << retorno.anoEntrada << " "
236                               << retorno.consumo << " " << endl;
237         } if(clienteEntrada->cpf == retorno.cpf) {
238             clientesARQSaida << clienteEntrada->nome << " "
239                               << clienteEntrada->cpf << " "
240                               << clienteEntrada->senha << " "
241                               << clienteEntrada->idade << " "
242                               << clienteEntrada->profissao << " "
243                               << clienteEntrada->nacionalidade << " "
244                               << clienteEntrada->sexo << " "
245                               << clienteEntrada->endereco << " "
246                               << clienteEntrada->diaEntrada << " "
```

Linha 170, Coluna 5 Caracteres 11616

INSERIR UTF-8 C++

Código onde foi implementado a função excluir, onde é pedido o CPF da pessoa a ser excluída , comparando esse CPF com uma série de CPFs lido do arquivo, e se esse CPF for igual ele não irá ser escrito em um novo arquivo.

```

179 void excluirCliente(string cpf) { //linha é o número da linha que vai ser deletada
180     fstream clientesARQ;
181     ofstream clientesARQSaida;
182     cliente retorno;
183     string linha;
184
185     clientesARQ.open("clientes.dat", ios::in);
186     fstream excluirCliente::clientesARQ
187     clientesARQSaida.open("clientesCopia.dat", ios::out | ios::trunc);
188
189     while (getline(clientesARQ, linha))
190     {
191         separarValoresClientes(linha, retorno);
192         if(cpf != retorno.cpf) {
193             clientesARQSaida
194                 << retorno.nome << ","
195                 << retorno.cpf << ","
196                 << retorno.senha << ","
197                 << retorno.idade << ","
198                 << retorno.profissao << ","
199                 << retorno.nacionalidade << ","
200                 << retorno.sexo << ","
201                 << retorno.endereco << ","
202                 << retorno.diaEntrada << ","
203                 << retorno.mesEntrada << ","
204                 << retorno.anoEntrada << ","
205                 << retorno.consumo << ";" << endl;
206         }
207
208     clientesARQ.close();
209     clientesARQSaida.close();
210
211     system("MOVE clientesCopia.dat clientes.dat");
212
213 }

```

Conclusão:

Esse projeto com plena certeza foi o maior desafio até agora, principalmente na parte de juntar os códigos dos outros grupos com o nosso. Mas fora isso, foi de grande aprendizado porque utilizamos novas estruturas como: ponteiros, namespaces, bibliotecas, o uso de interfaces e outros tópicos.

Esse sistema também nos permitiu ver na prática como é trabalhar com várias pessoas, produzindo conteúdos que faria parte de um todo no futuro e, que seria no futuro também a parte mais complexa.