Documentação Técnica e Funcional do Software de Gerenciamento de Hotel

João Paulo de Souza Marco Túlio Nogueira Cambraia

11 de Dezembro de 2017

Sumário

1 Estruturação das Interna Pastas

Os arquivos são estruturados na pasta interna software de forma que seguem a seguinte ordenação e nomeclatura:

1.1 Pasta 'libs'

Guarda os arquivos com extensão .h, que são utéis no funcionamento do software. A funcionalidade específica ou fluxo de dados será explicada mais a frente. Segue a lista dos arquivos de cabeçalho presentes na pasta citada:

1. **structs.h:** O arquivo de cabeçalho structs.h, trata somente de organizar as estruturas ou 'structs' presentes e necessárias no código fonte.;

2. **cadastros.h:** O arquivo de cabeçalho cadastros.h, trata de utilizar as estruturas, citadas no item acima, para armazenar os dados digitados pelo usuário em certas ocasiões do funcionamento do software e realizar algumas pequenas validações básicas utéis no código, como por exemplo, verificar o que o usuário deseja entre duas opções. Tudo isso irá armazenar dados que serão necessários em outros arquivos de cabeçalho que irão ser citados a frente.

3. **salvar.h:** Este arquivo de cabeçalho serve para de que depois que os dados que os dados forem armazenados pela junção dos dois arquivos citados acima, que as informações sejam reunidas e salvas em um arquivo no formato texto(txt) ou em formato binário. Neste arquivo de cabeçalho, são salvos os dados são salvos alguns dados espéficos, ou dados mais comuns, que são os dados que envolvem diretamente atividades comuns de um sistema, exemplo salvar um hospede cadastrado. Estruturas mais complexas, tem seus próprios arquivos de cabeçalho para gerenciar seus dados.

4. **consulta.h:** Este arquivo header(cabeçalho), serve para realizar alguns tipos de consultas nos arquivos que são salvos pelo software. Exemplo de umas das funcionalidades é gerar códigos auto-incrementados(códigos somados de forma automática) para o uso das structs. Outro exemplo seria a consulta de dados de structs simples, como hospede e/ou produto. E por fim mais uma de suas funcionalidades, funções de validação e de pesquisa por certos campos, exemplo, código do hospede de acordo com seu CPF.

5. **edicao.h:** Este arquivo header serve novamente para structs simples do software, como produtos e/ou fornecedores. Nela estão contidas funções

para editar dados dos arquivos pertencentes as essas structs citadas anteriormente. Nessas funções é possível selecionar o que será editado, mas a preço de editar um campo por vez.

6. **exclusao.h:** Neste arquivo header, estão contidas as funções de exclusão das structs simples do software, a função tem como funcionamento base, receber o código do daquilo que será excluído e partir disso localizar o dado de acordo com o código e elimina-lo do arquivo.

7. **cores.h:** Este arquivo de cabeçalho serve para possibilitar a digitação de texto dentro do software com cores diferentes. Como função pré-estabelecida, a conversão de cores funciona apenas para digitação que esteja 100% em formato texto, podendo ser utilizadas algumas cores básicas e outras secundárias e terciárias, o funcionamento interno específico será explicado com mais afim futuramente dentro deste documento.

8. **config.h:** Este é um dos arquivos de cabeçalhos mais importantes do software, ele salva os dados de configuração base do usuário, como o tipo de salvamento do software, podendo alterar entre texto e binário e gera o login e senha master do sistema, que serão o primeiro acesso do sistema, antes mesmo de houver usuários cadastrados.

9. **produto.h:** Este arquivo de cabeçalho, seria um dos arquivos mais complexos, citados acima. Suas funções tem como propósito controlar o fluxo de entrada e saída de produtos dentro do hotel, existindo funções para o cadastro de notas de entradas de produtos e funções para atualização de preços de venda e quantidade de produtos após cada venda ou compra.

10. **reserva.h:** Este seria outro arquivo de cabeçalho dos complexos, suas funções tem como objetivo, cadastrar, gerenciar e realizar consultas e verificações sobre as possíveis reservas que os hospedes venham a realizar no hotel, permitindo até mesmo o hospede pesquisar a acomodação que contenha as facilidades que deseja.

11. **checks.h:** Outro arquivo de cabeçalho considerado complexo, seus objetivos principais são gerenciar a realização de check in e check out por parte do hospede, permitindo a ele pagar antecipadamente ou não a sua permanência no hotel, não incluíndo seus gastos com produtos vendidos no hotel.

12. **contas.h:** Mais um dos arquivos de cabaçalho complexos, suas funções tem como propósito base, cadastrar, gerenciar gastos e fazer o fechamento(pagamento)

das contas dos hospedes. O hospede pode anotar os produtos que consume dentro do hotel e na realização do check out, deverá pagar segundo a sua conta informar.

13. **importacao_exportacao.h:** Arquivo de cabeçalho complexo, armazena todos os dados dos arquivos presentes localmente no software e os agrupa em formato xml para que possa ser exportado e utilizado em outros softwares. Sua funcionalidade de importação infelizmente não foi finalizada, provavelmente posteriormente será finalizada e será utilizável.

14. **caixa.h:** Arquivo de cabaçalho complexo e muito importante, suas funções tem como objetivo gerenciar a abertura do caixa, de seu fluxo de entrada e saída de dinheiro e principalmente, quando não houver dinheiro suficiente em caixa para o pagamento de uma dívida, enviar os dados para contas a pagar, que futuramente serão pagas assim que o caixa estiver com dinheiro suficiente para isso, ou seja não permite o caixa com valores negativos.

15. **feedback.h:** Outro arquivo header complexo, infelizmente, não houve tempo para sua implementação, mas provavelmente, posteriormente será implementado e será utilizável dentro do software.

1.2 Pasta 'saves'

O funcionamento desta pasta é simples, ela é o diretório padrão onde todos os arquivos, sejam eles originais ou temporários são e serão salvos.

1.3 Pasta 'config'

Outro diretório de funcionamento e objetivo simples, neste basicamente é guardado o arquivo de configuração do usuário, onde contém seu modo de salvamento, sendo ele texto ou binário e o login e senha master.

2 Funções e seus objetivos

Abaixo serão listados todos as funções presentes no código e seus objetivos, explicados de uma forma simples
Obs: As funções como muitos parametros, geralmente necessitam de um pouco mais de validação externa, ou seja vinda de outros arquivos.

2.1 Cadastros.h

1. **struct hospede cadastrahospede():** Função que recebe os dados do hospede que será cadastrado, armazena na struct correspondente e retorna essa struct para ser utilizada por outras funções.

2. **struct hotel cadastrahotel():** Função que recebe os dados do hotel que será cadastrado, armazena também na struct e retorna para ser utilizado ao decorrer do código.

3. **struct categorias cadastracategoria():** Função que recebe os dados da categoria(grau da acomodação) que será cadastrada, armazena na struct e retorna para ser utilizado ao decorrer do código.

4. **struct acomodacoes cadastraacomodacao(char urlcategoria[50],char modoabertura[5]):** Função que recebe os dados da acomodação que será cadastrada, armazena na struct e retorna para ser utilizado ao decorrer do código. Recebe por parametro o caminho do arquivo de categoria e o modo de abertura do arquivo.

5. **struct produtos cadastraproduto():** Função que recebe os dados do produto que será cadastrado, armazena também na struct e retorna para ser utilizado em outras funções.

6. **usuarios cadastrausuario():** Função que recebe os dados do produto que será cadastrado, armazena também na struct e retorna para ser utilizado em outras funções. E por motivos de redundância, outras funções com apenas objetivo de cadastro de dados nas struct não terão seu objetivo listado

7. **struct fornecedores cadastrafornecedor():** Função para cadastrar fornecedores

8. **struct entradaprodutos cadastra_entradaprodutos(char urlproduto[50],char urlfornecedor[50],char urlcaixa[50],char urltempcaixa[50], char urlctp[50],char modoabertura[5]):** Função para cadastro de entrada de produtos

9. **struct saidaprodutos cadastra_saidaprodutos(char urlproduto[50],char urlcontas[50],char urltempcontas[50],char urlcaixa[50],char urltempcaixa[50], char urlhospede[50],char urlctp[50],char modoabertura[5]):** Função para cadastro de saída de produtos

10. **struct reserva cadastra_reserva(char urlacomodacao[50],char urlcategoria[50],char urlhospede[50],char modoabertura[5]):** Função para o cadastro de reservas

11. **void pesquisa_acomodacao(char urlacomodacoes[50],char urlcategoria[50],char urlreserva[50],char modoabertura[5]):** Esta função recebe os dados da preferência do hospede em uma acomodação e de acordo com isso tenta encontrar um acomodação vaga que preencha esses requisitos, ou pelo menos tentar dependendo do requisito.

12. **struct checks checagens(char urlacomodacao[50],char urlcategoria[50],char urlhospede[50],char urlreserva[50],char urlchecks[50],char urlcontas[50], char urltempcontas[50],char urlcaixa[50],char urltempcaixa[50], char urlctp[50], char modoabertura[5]):** Esta função realiza o armazenanto de informações cruciais para a realização de check in ou check out dentro do software.

13. **struct contas cadastra_conta(char urlconta[50],char urltempconta[50],char urlhospede[50],char urlreserva[50],char modoabertura[5]):** Outra função de cadastro, mas para contas dos hospedes.

14. **struct caixa cadastra_caixa(char urlhotel[50], char modoabertura[5]):** Outra função de cadastro, só que agora para caixa.

15. **struct caixa abertura_fechamento_caixa(char urlhotel[50],char urlcaixa[50], char modoabertura[5],int tipo): Função que gerencia os dados necessário para abertura e fechamento de caixa**

2.2 salvar.h

Abaixo será listado as funções pertencentes a esse arquivo de cabeçalho:

1. **void salvarhospede(int tipo,char url[50],char modoabertura[5],struct hospede h):** Função que realiza o salvamento dos dados do hospede recibidos da struct e da função de cadastro, o salvamento é feito em arquivo texto ou binário.

2. **void salvarhotel(int tipo,char url[50],char modoabertura[5],struct hotel ht):** Função que realiza o salvamento dos dados do hotel, vindo da struct e da função de cadastro, o salvamento também pode ser feito em arquivo texto ou binário.

3. **void salvarcategorias(int tipo,char url[50],char modoabertura[5],struct categorias c):** Função para realizar o salvamento de categorias, ou dados são vindos da struct de categorias, manipulada dentro da função de cadastro, os dados podem ser salvos em texto ou binário.

4. **void salvaracomodacao(int tipo,char url[50],char modoabertura[5],struct acomodacoes ac):** Função para o cadastro dos dados de acomodação, podendo ser em texto ou binário, os dados vem da função de cadastros trabalhando com a struct.

5. **void salvarproduto(int tipo,char url[50],char modoabertura[5],struct produtos p):** Função para o cadastro de produtos, recebe os dados da função de cadastro em conjunto com a struct e salva os dados em texto ou binário.

6. **void salvarfornecedor(int tipo,char url[50],char modoabertura[5],struct fornecedores f):** Função de cadastro de fornecedores, recebe os dados da função de cadastro em conjunto com a struct e salva os dados em arquivo texto ou binário.

7. **void salvarusuarios(char url[50],char modoabertura[5],struct usuarios u):** Função para cadastro de usuários, os usuários são cadastrados apenas em binário por questão de segurança. Os dados para o cadastro vem da função de cadastro em conjunto com a struct.

2.3 Consulta.h

1. **void consultahospede(int tipo,char url[50],char modoabertura[5]):** Função para consultar todos os hospedes cadastrados e apresenta-los em tela, em forma de lista para o usuário.

2. **void consultahotel(int tipo,char url[50],char modoabertura[5]):** Função para consultar os dados de todos os hoteis cadastrados seja em texto ou binário e apresenta-los em tela na forma de lista para o usuário.

3. **void consultaproduto(int tipo,char url[50],char modoabertura[5]):** Função para a realizar a consulta de produtos e mostrar os dados em lista na tela.

4. **void consultaacomodacao(int tipo,char url[50],char modoabertura[5]):** Função para a consulta dos dados das acomodações cadastradas, mostra os dados na tela em forma de lista.

5. **void consultacategoria(int tipo,char url[50],char modoabertura[5]):** Função para a consulta de dados cadastrados das categorias, é mostrado em forma de lista para o usuário.

6. **void consultafornecedor(int tipo,char url[50],char modoabertura[5]):** Função para a consulta dos dados cadastrados de fornecedores, mostra os dados na tela em forma de lista para o usuário.

7. **void consultausuario(char url[50],char modoabertura[5]):** Função que consulta os dados dos usuários cadastrados, mostra em forma de lista na tela os dados.

   Obs:O restante das funções a seguir dispensam explicação pois são funções de validação de código e de auto incrementação de código.

8. **int codigohospede(int tipo)**
9. **int codigohotel(int tipo)**
10. **int codigoproduto(int tipo)**
11. **int codigoacomodacao(int tipo)**
12. **int codigocategoria(int tipo)**
13. **int codigofornecedor(int tipo)**
14. **int codigousuario()**
15. **int verificausuario(char login[20],char senha[20]):** Função para validar a existencia do login e de sua senha correspondente, se sim, permite o usuário logar no sistema.

16. **int codigo_entradaprodutos(int tipo)**
17. **int codigo_saidaprodutos(int tipo)**
18. **int codigoreserva(int tipo)**
19. **int codigo_checks(int tipo)**
20. **int codigo_conta(int tipo)**
21. **int codigo_caixa(int tipo)**
22. **int codigo_contapagar(int tipo)**
23. **int valida_codigohospede(int tipo,char url[50],char modoabertura[5],int codigo)**
24. **int valida_codigohotel(int tipo,char url[50],char modoabertura[5],int codigo)**
25. **int valida_codigoproduto(int tipo,char url[50],char modoabertura[5],int codigo)**
26. **int valida_codigocategoria(int tipo,char url[50],char modoabertura[5],int codigo)**
27. **int valida_codigoacomodacao(int tipo,char url[50],char modoabertura[5],int codigo)**
28. **int valida_codigofornecedor(int tipo,char url[50],char modoabertura[5],int codigo)**
29. **float retorna_valoracomodacao(int tipo,char urlacomodacao[50],char urlcategoria[50],char modoabertura[5],int codigo):** Função que retorna o valor da acomodação de acordo com seu código, que é passado por parametro.
30. **int* retorna_dia_checkin_pago(int tipo,char url[50],char modoabertura[5],int codigo):** Função que retorna um ponteiro de vetor, o vetor contém o dia do check e se o valor da diaria foi paga pelo hospede.
31. **int codigo_hospede_cpf(int tipo,char urlhospede[50],char modoabertura[5],char cpf[14]):** Função que retorna o código do hospede de acordo com o cpf digitado, esse cpf é recebido por parametro.
32. **int codigo_hotel_cnpj(int tipo, char url[50], charmodoabertura[5], char cnpj[20]):** Função que retorna o código do hotel de acordo com o cnpj digitado, esse cnpj é recebido por parametro.

33. **int verifica_existencia_arquivo(int tipo,char urlarquivo[50]):** Função que verifica a existencia de um arquivo pelo seu url.

34. **int verifica_vazio(int tipo, char urlarquivo[50]):** Função que verifica se o arquivo se encontra vazio, pelo seu tamanho a partir de seu url.

2.4 edicao.h

1. **void editahospede(int tipo,char url[50],char modoabertura[5],char urltemp[50]):** Função que edita os dados dos hospedes, o usuário pode escolher o dado a ser editado, mas pode editar apenas um dado por vez. Funciona para os dois tipo de arquivo, texto e binário.

2. **void editahotel(int tipo,char url[50],char modoabertura[5],char urltemp[50]):** Função para editar os dados dos hoteis, podem ser editados qualquer dado do hotel, menos seu código e desde que seja editado um dado por vez.

3. **void editaproduto(int tipo,char url[50],char modoabertura[5],char urltemp[50]):** Função para edição de dados do produto, ainda não foi finaliza, mas quando for fara o mesmo que as outras e seguirá a mesma forma.

4. **void editacategoria(int tipo,char url[50],char modoabertura[5],char urltemp[50]):** Função para editar os dados de categoria, desde que seja um dado por vez.

5. **void editaacomodacao(int tipo,char url[50],char modoabertura[5],char urltemp[50],char urlcategoria[50]):** Função para editar os dados da acomodação, permite editar apenas um dado por vez, serve para os dados salvos tanto em texto quanto em binário.

6. **void editafornecedor(int tipo,char url[50],char modoabertura[5],char urltemp[50]):** Função para editar os dados do fornecedor, um dado por vez, mas qualquer dado que não seja o código.

7. **void editausuario(char url[50],char modoabertura[5],char urltemp[50]:** Função para editar os dados do usuário, edita um dado por vez, e por editar qualquer dado que não seja o código.

2.5 exclusao.h

1. **void excluihospede(int tipo,char url[50],char modoabertura[5],char urltemp[50]):** Função para excluir os dados de um hospede específico. Lê e mostra todos e pergunta para o usuário qual deles será excluído.

2. **void excluihotel(int tipo,char url[50],char modoabertura[5],char urltemp[50]):** Função para excluir dados de um hotel. Lê os hoteis cadastrados e mostra, em seguida pergunta para o usuário qual ele quer excluir.

3. **void excluicategoria(int tipo,char url[50],char modoabertura[5],char urltemp[50]):** Função para excluir alguma categoria. Mostra as categorias cadastradas, em seguida o usuário decide qual excluir.

4. **void excluiacomodacao(int tipo,char url[50],char modoabertura[5],char urltemp[50]):** Função para excluir alguma acomodação. Mostra as acomodações cadastradas, em seguida o usuário decide uma para ser excluida.

5. **void excluiproduto(int tipo,char url[50],char modoabertura[5],char urltemp[50]):** Função para excluir um produto. Mostra os produtos cadastrados, e o usuário decide qual será excluído.

6. **void excluifornecedor(int tipo,char url[50],char modoabertura[5],char urltemp[50]):** Função para exclur um fornecedor. O usuário verá todos os fornecedores cadastrados e decidirá um para ser excluído.

7. **void excluiusuario(char url[50],char modoabertura[5],char urltemp[50]):** Função para excluir um usuário, a partir dos que estão cadastrados, assim o usuário decide um para ser excluído.

8. **void resetaconfig():** Função para resetar as configurações de salvamento e login master.

2.6 produto.h

1. **float* entrada_produtos(int tipo,char url[50],char modoabertura[5],struct entradaprodutos ep):** Função que após realizar a o cadastro da entrada de produtos, retorna o valor do imposto e do frete em um vetor, que a partir de um ponteiro poderá ser utilizado em outra função.

2. **void atualiza_valorprodutos(int tipo,char url[50],char modoabertura[5],char urltemp[50],float *valores,struct entradaprodutos ep):** Função que atualiza os valores dos produtos que foram comprados pelo imposto e valor de frete, que foram retornados na função acima.

3. **void atualiza_quantidadeprodutos(int tipo,char url[50],char urltemp[50],char modoabertura[5],struct saidaprodutos sp):** Função para atualizar a quantidade de produtos após um venda.

4. **void saida_produtos(int tipo,char url[50],char modoabertura[5],struct saidaprodutos sp):** Função para o cadastro de uma venda de produtos. Os produtos assim como na compra, podem ser variados e conter mais de um cada.

5. **float retorna_valoresprodutos(int tipo, char url[50], char modoabertura[5],int codigo):** Função para retornar o valor do produto de acordo com o código digitado, esse que é recebido por parametro.

6. **relatorio_vendas(int tipo,char url[50],char modoabertura[5]):** Função que lê os dados de todos as vendas, faz a soma total disso e mostra na tela em forma de relatório.

2.7 reserva.h

1. **void reserva(int tipo,char url[50],char modoabertura[5],struct reserva r):** Função que realiza a reserva do hospede, a partir dos dados que vem pelo struct.

2. **void consulta_reserva(int tipo,char url[50],char modoabertura[5],int codigo):** Função que consulta uma reserva de acordo com seu código.

3. **void consulta_reservas(int tipo,char url[50],char modoabertura[5]):** Função para consultar e mostrar todas as reservas até o momento.

4. **void cancela_reserva(int tipo,char url[50],char urltemp[50],char modoabertura[5],int codigo):** Função para o cancelamento de reserva, a partir de seu código.

5. **int* pesquisa_acomodacoes(int tipo,char urlacomodacoes[50],char urlcategoria[50],char urlreserva[50],char modoabertura[5], struct**

data d,int quantidade_adultos,int quantidade_criancas,int tv,int tvcabo,int arcondicionado, int frigobar,int banheiro,int camacasal,int camasolteiro,int banheira,int hidromassagem): Função de pesquisa de acomodação por requisitos, verifica quais são os requisitos do usuário e filtra as acomodações a partir disso.

2.8 checks.h

1. **void check(int tipo,char url[50],char urltemp[50],char modoabertura[5],struct checks ch):** Função que realiza o check, independente se for check in ou check out.

2. **void consulta_checks(int tipo,char url[50],char modoabertura[5]):** Função para consultar os checks realizados até o momento.

3. **void consulta_checks_codigo(int tipo,char url[50],char modoabertura[5],int codigo):** Função que consulta o check, de acordo com o código que veio por parametro.

2.9 contas.h

1. **void conta_hospede(int tipo,char url[50],char urltemp[50],char modoabertura[5],struct contas ct,int funcao):** Função que gerencia a conta do hospede, tem 4 funções básicas, cadastra conta novas, consultar todas as contas cadastradas, inserir um valor na conta, simulando uma compra e por fim fechar a conta no check out.

2.10 config.h

1. **void configsave(int op,char mlogin[20],char msenha[20]):** Função que faz a configuração inical do sistema, recebendo a opção de salvamento, o login master e sua senha.

2. **int verificasave():** Função que verifica o tipo de salvamento cadastrado.

2.11 importacao_exportacao.h

1. **void exporta_tabelas(int tipo,char modoabertura[5]):** Função que recebe quais as tabelas vão ser exportadas, e o nome do arquivo. Vai verificando arquivo a arquivo, para ver qual será exportado, permitindo

assim exportar mais de um. Salva o arquivo na área de trabalho com o nome desejado em formato xml.

2.12 main.c

1. **void defineconstantes():** Função que define o caminho das urls originais e temporárias de todos os arquivos de acordo com o tipo de salvamento cadastrado.

2. **int login():** Função para verificação e validação do login e da senha.

3. **void config():** Função que realiza a coleta dos dados para a configuração inicial.

4. **int verificaconfig():** Função para verificar o tipo de configuração que foi cadastrada.

5. **void menu(char com[50],int tiposave):** Função de menu, permite ao usuário digitar o comando que ele deseja executar.

3 help(ajuda)

Abaixo serão listados todos os comando do sistema e suas funcionalidades.
Obs: Os comandos serão mostrados da forma que são escritos no sistema.

3.1 Comando relacionados ao hospede

•cadhp - Comando para o Cadastro de Hospedes.
•cshp - Comando para o Consulta de Hospedes.
•edhp - Comando para o Edição de Dados de Hospedes.
•exhp - Comando para o Exclusão de Dados de Hospedes.

3.2 Comando relacionados ao hotel

•cadht - Comando para o Cadastro de Hoteis.
•csht - Comando para o Consulta de Hoteis.
•edht - Comando para o Edição de Dados de Hoteis.
•exht - Comando para o Exclusão de Dados de Hoteis.

3.3 Comando relacionados ao produto

•cadp - Comando para o Cadastro de Produtos.
•csp - Comando para o Consulta de Produtos.
•edp - Comando para o Edição de Dados de Produtos.
•exp - Comando para o Exclusão de Dados de Produtos.

3.4 Comando relacionados ao fornecedor

•cadf - Comando para o Cadastro de Fornecedores.
•csf - Comando para o Consulta de Fornecedores.
•edf - Comando para o Edição de Dados de Fornecedores.
•exf - Comando para o Exclusão de Dados de Fornecedores.

3.5 Comando relacionados ao usuário

•cadus - Comando para o Cadastro de Usuários.
•csus - Comando para o Consulta de Usuários.
•edus - Comando para o Edição de Dados de Usuários.
•exus - Comando para o Exclusão de Dados de Usuários.

3.6 Comando relacionados a categoria

•cadc - Comando para o Cadastro de Categorias.
•csc - Comando para o Consulta de Categorias.
•edc - Comando para o Edição de Dados de Categorias.
•exc - Comando para o Exclusão de Dados de Categorias.

3.7 Comando relacionados a acomodação

•cadac - Comando para o Cadastro de Acomodações.
•csac - Comando para o Consulta de Acomodações.
•edac - Comando para o Edição de Dados de Acomodações.
•exac - Comando para o Exclusão de Dados de Acomodações.

3.8 Comando relacionados a venda/compra de produtos

•cpp - Comando Compra de Produtos.
•vdp - Comando Venda de Produtos.

3.9 Comando relacionados a reserva

•rsv - Comando para a Realizar a Reserva.
•csrsvc - Comando para o Consultar a Reserva de acordo com o Código.
•csrsv - Comando para o Consultar as Reservas.
•canrsv - Comando para Cancelar a Reserva.
•psrsv - Comando para Pesquisar uma Acomodação de acordo com as Facilidades.

3.10 Comando relacionados aos checks

•chk - Comando para a Realizar o Check in ou Check out.
•cschk - Comando para o Consultar os Checks.
•cschkc - Comando para o Consultar os Checks de acordo com o Código.

3.11 Comando relacionados a conta

•cadct - Comando para Cadastrar a Conta.
•csct - Comando para o Consultar a Conta de acordo com o CPF.

3.12 Comando relacionados a exportação/importação

•exparq - Comando para Exportar os arquivos.

3.13 Comando relacionados ao caixa

•cadcx - Comando para Cadastrar o Caixa.
•caixa - Comando para Abrir ou Fechar o Caixa.
•ctp - Comando para Inserir ou Pagar uma Conta a Pagar

3.14 Outros comandos

•lt - Limpa a Tela.
•rconfig - Reseta a Configuração Inicial.
•ctp - Comando para Inserir ou Pagar uma Conta a Pagar
•help - Mostra os Comandos do Sistema
•sair - Comando para Sair do Sistema

4 Funcionamento Básico

4.1 Tela de Login

A figura mostra inicialmente a Tela de Login, onde aparecerá para o usuário digitar seu login e senha

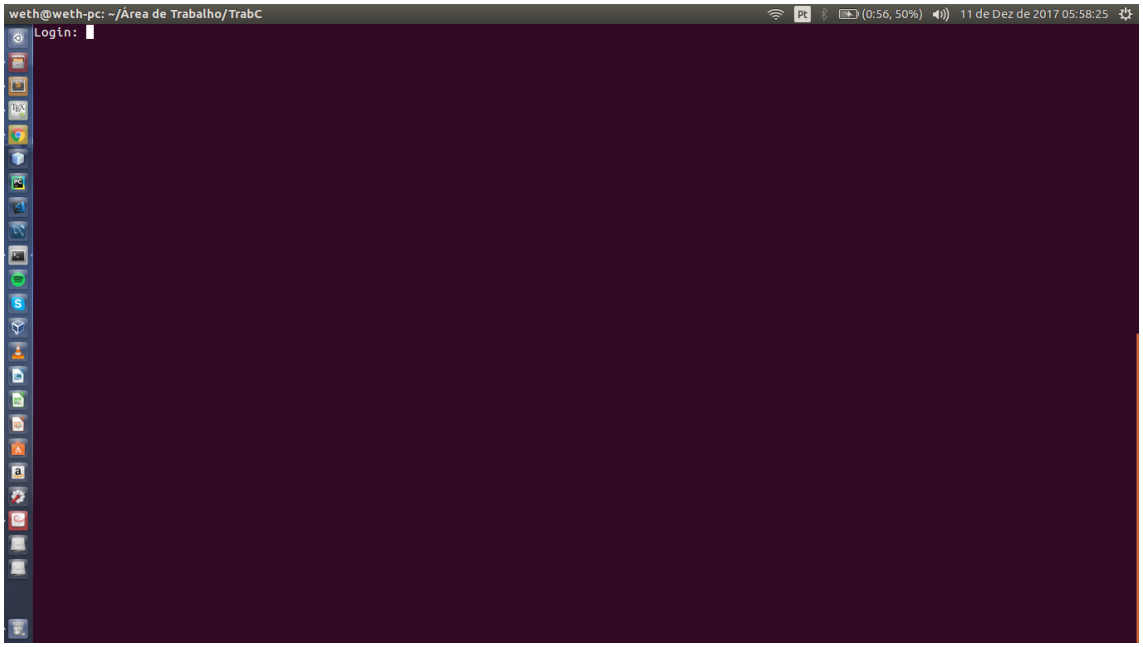

Figura 1: Imagem da Tela de Login

Se o login e a senha estiverem corretos, aparecerá uma mensagem assim

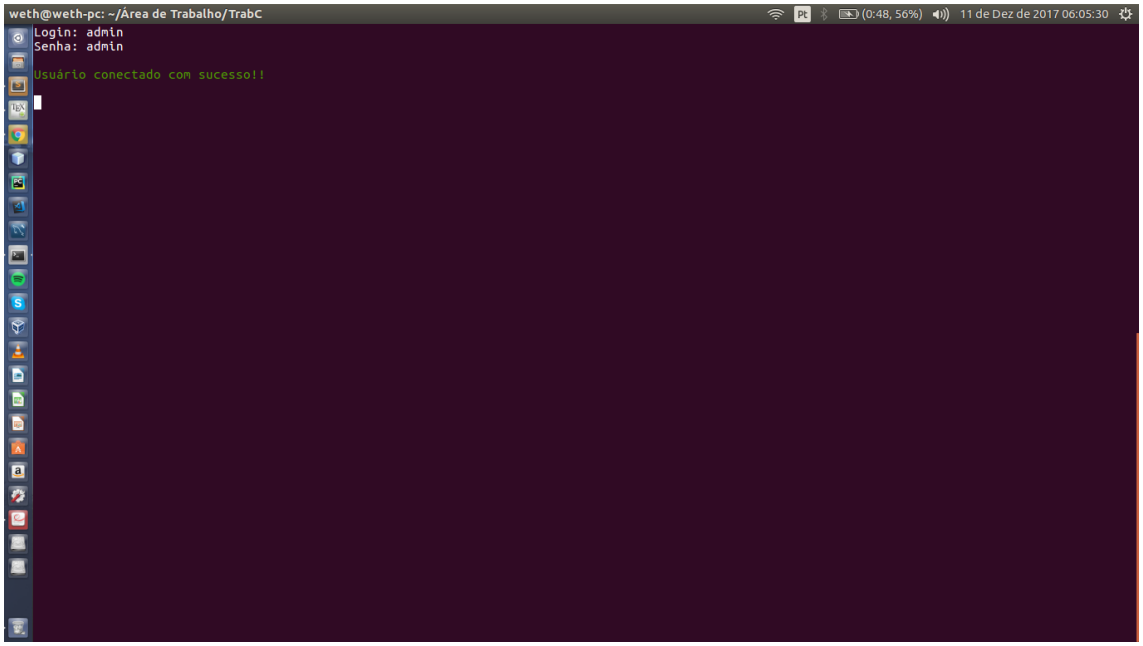

Figura 2: Imagem2 da Tela de Login

Outro procedimento seria em digitar um comando, no exemplo será o comando de cadastro de hospedes

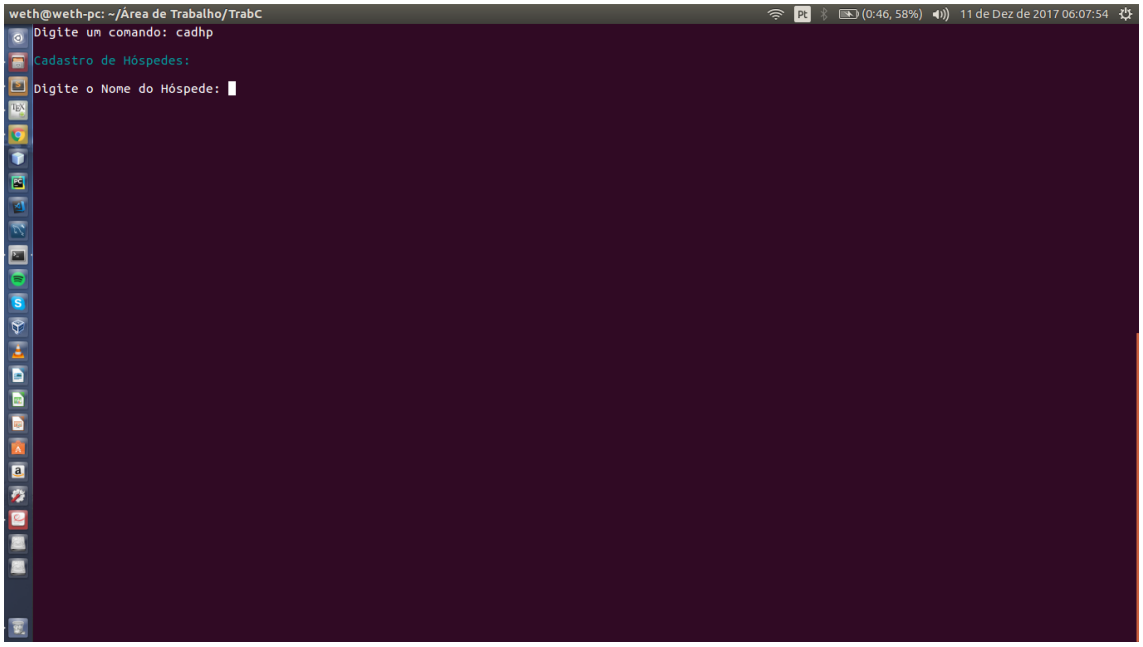

Figura 3: Imagem da Tela do Sistema, onde foi digitado o comando de Cadastro de Hospedes

Após serem digitados todos os dados(nesse caso dados hipotéticos), será mostra uma mensagem de sucesso(verde - sucesso, amarelo - atenção, vermelho - perigo/gravidade, azul-claro - legenda.)

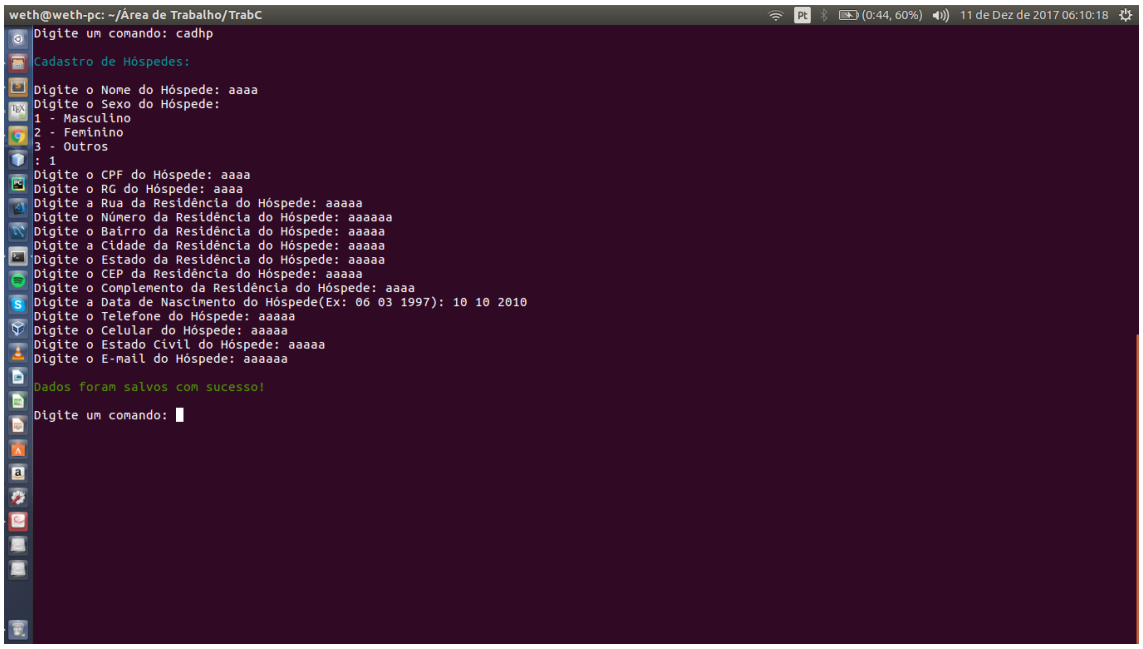

Figura 4: Imagem2 da Tela do Sistema, onde foi digitado o comando de Cadastro de Hospedes