Documentação - Trabalho de AEDS I e Fundamentos de Engenharia de Software
Luis Henrique Barbosa e Marcela Mendes Campos

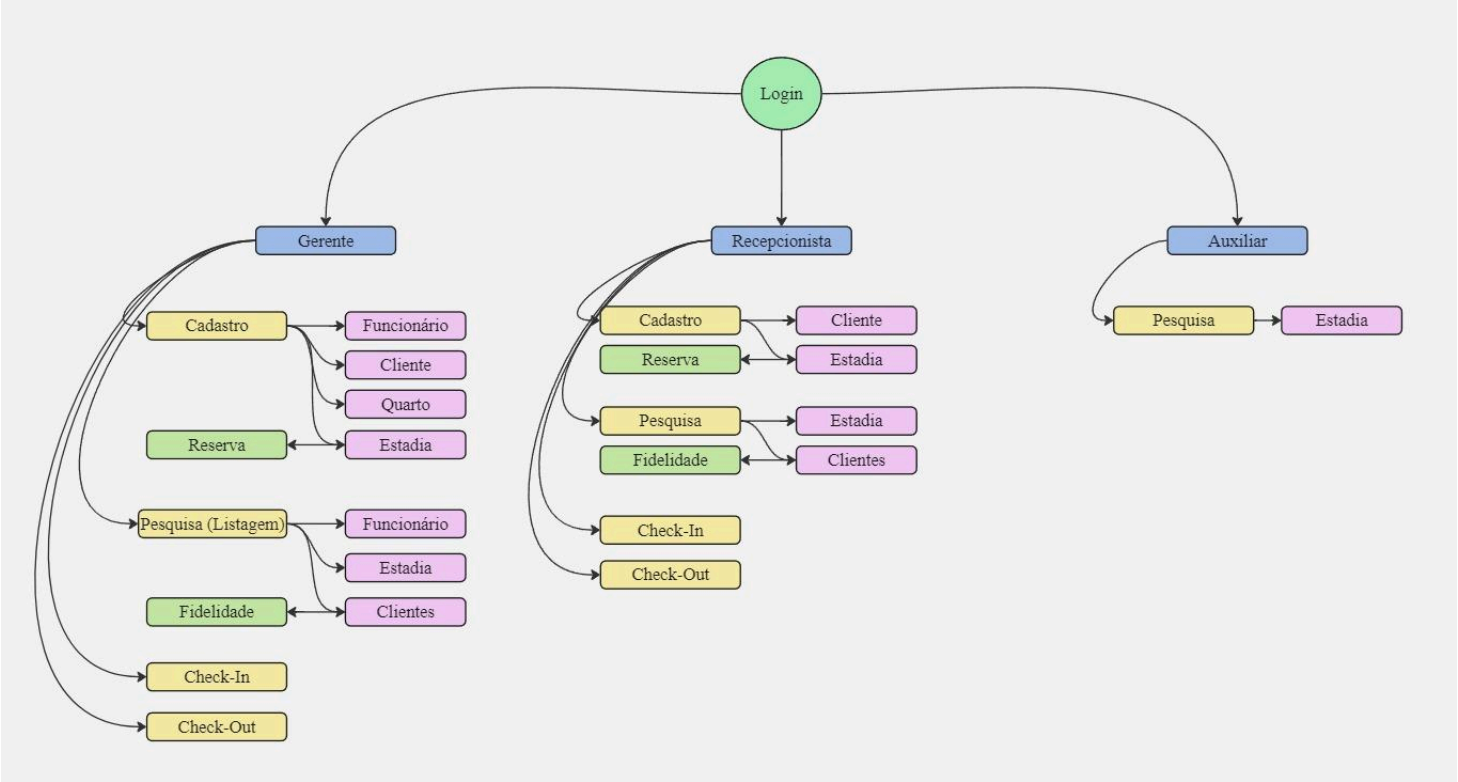
Software de Menu para Hotel Descanso Garantido

O objetivo do projeto é migrar um sistema de gestão de hotel anteriormente baseado em planilhas do Excel e cadernos para um software integrado. Este novo sistema permitirá aos funcionários do hotel realizar todas as funções essenciais, como cadastros de clientes, gestão de reservas, realização de hospedagens, cálculo de pontos de fidelidade, e outras operações de forma automatizada e centralizada.

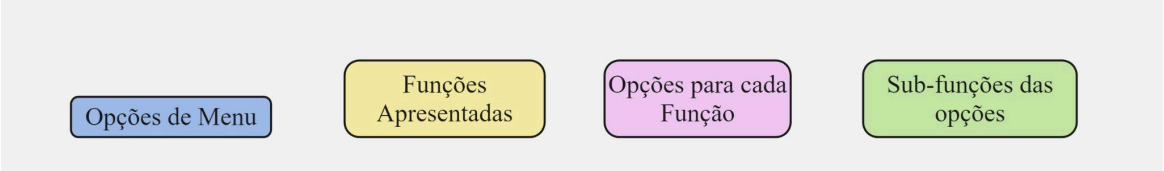
Arquitetura da Solução

A ideia para o software foi de criar uma tela inicial de Login, permitindo que os funcionários loguem e automaticamente o sistema redireciona para um menu específico com funções específicas para cada funcionário.

1- Fluxograma para Representação Básica do Software



Legenda:



Logins Para Uso do Sistema

Gerente:	**Login:** 1	**Senha:** 123123
Auxiliar de Limpeza:	**Login:** 2	**Senha:** 123123
Recepcionista:	**Login:** 3	**Senha:** 123123
Garçom:	**Login:** 4	**Senha:** 123123

Stack da Solução

A linguagem de programação usada para o desenvolvimento foi somente C. E o trabalho foi feito inteiramente no Replit.com. Foram usadas ferramentas auxiliares como Trello e Miro para a documentação, divisão de tarefas e para ajudar a desenvolver a ideia do Programa.

Descrição Geral do Código e Arquivos

O projeto está organizado com o arquivo main.c, acompanhado de um par de arquivos .h e .c para cada entidade principal do sistema: estadia, quarto, funcionário e cliente. No contexto desse projeto, os arquivos .h servem como um arquivo de cabeçalho no projeto, proporcionando uma interface que expõe as estruturas de dados e as assinaturas das funções para outros arquivos .c que precisam utilizá-las. E os arquivos .c é onde estão implementadas as funções declaradas no arquivo de cabeçalho

Estrutura dos Dados Principais do Programa

1. **MAIN.C**

O arquivo main.c possui uma lista de procedimentos **CriaSeNaoExisteCliente(); CriaSeNaoExisteQuarto(); CriaSeNaoExisteFuncionario(); CriaSeNaoExisteEstadia();** (que serão especificadas em 2.a,).

Além disso, abre o login para identificação do usuário usando o programa e redireciona o usuário para o procedimento específico do menu designado a tal usuário.

2. **CLIENTE.C**
    a. **void CriaSeNaoExisteCliente()**

Procedimento que verifica se o arquivo de clientes (PATH_CLIENTE) existe. Se não existir, ela cria o arquivo e escreve um conjunto inicial de clientes nele. Se o arquivo já existir, ela simplesmente fecha o arquivo sem fazer alterações.

b. void ListaCliente()

Esta função lista todos os clientes cadastrados no arquivo PATH_CLIENTE. Ela abre o arquivo para leitura ("rb"), lê cada cliente um por um utilizando fread() e imprime suas informações formatadas na tela.

c. int quantosClientes()

Esta função conta quantos clientes estão cadastrados no arquivo PATH_CLIENTE. Ela também abre o arquivo para leitura ("rb"), conta quantos registros de cliente existem usando um contador simples e retorna esse número.

d. void CadastraCliente()

Esta função permite cadastrar um novo cliente no sistema. Ela solicita ao usuário que insira o nome, telefone e endereço do cliente. Após validar os dados, gera automaticamente um código para o novo cliente (somando 1 ao total de clientes existentes) e salva essas informações no arquivo PATH_CLIENTE em modo de escrita ("ab").

e. void PesquisaClientePorCodigo()

Esta função permite pesquisar um cliente pelo seu código. Ela solicita ao usuário que insira o código do cliente desejado e, em seguida, busca esse cliente no arquivo PATH_CLIENTE. Se encontrar, imprime suas informações formatadas na tela.

f. void PesquisaClientePorNome()

Esta função permite pesquisar clientes cujo nome contenha uma determinada substring fornecida pelo usuário. Ela solicita ao usuário que insira a substring desejada e busca todos os clientes cujos nomes contenham essa substring no arquivo PATH_CLIENTE. Para cada cliente encontrado, imprime suas informações formatadas na tela.

g. bool ClienteExiste(int codigo)

A função ClienteExiste verifica se um cliente com um código específico existe no arquivo de dados dos clientes (clients.dat). Ela abre o arquivo, percorre os registros de clientes, e compara o código fornecido com o código de cada cliente armazenado. Se encontrar um cliente com o código especificado, retorna true. Caso contrário, retorna false.

3. ESTADIA.C

a. int transformaEmDias(struct Data data)

A função transformaEmDias converte uma data representada por uma estrutura Data (que contém os campos ano, mes e dia) no número total de dias desde o "início do tempo" (considerando um cálculo simplificado). O cálculo assume que todos os anos têm 365 dias e

todos os meses têm 30 dias, o que não leva em conta anos bissextos ou a variação real no número de dias dos meses.

b. void CriaSeNaoExisteEstadia()

Essa função é responsável por criar o arquivo de estadias se ele não existir. Ela abre o arquivo em modo leitura ("rb") e, se o arquivo não existir, abre novamente em modo escrita ("wb") para criar o arquivo vazio. Em seguida, fecha o arquivo.

c. int calcularDiario(struct Data inicio, struct Data fim)

A função calcularDiario calcula a diferença em dias entre duas datas, representadas pelas estruturas Data inicio e fim. Ela utiliza a função transformaEmDias para converter cada data no número total de dias desde um ponto inicial e então calcula a diferença entre esses dois valores.

d. bool PodeReserva(int codigoQuarto, struct tm inicio, struct tm fim)

Essa função verifica se um quarto pode ser reservado para as datas especificadas. Ela lê as estadias do arquivo de estadias, compara os períodos de estadia para o mesmo quarto e retorna verdadeiro se o quarto estiver disponível para reserva ou falso caso contrário.

e. int geraCodigo()

A função geraCodigo() é utilizada para gerar um novo código para uma estadia com base nos códigos existentes no arquivo de estadias. Ela conta quantas estadias já estão registradas, adiciona 1 ao último código encontrado e retorna esse novo código para ser usado na criação de uma nova estadia.

f. void RealizarReserva()

Essa função permite realizar a reserva de um quarto para um cliente com base nas entradas fornecidas pelo usuário. Ela solicita o código do cliente, número do quarto, número de hóspedes e datas de início e fim da estadia. Em seguida, verifica se o quarto pode ser reservado usando a função PodeReserva() e, se possível, cadastra a estadia no arquivo de estadias.

g. void ListaEstadia()

Essa função lista todas as estadias cadastradas no arquivo de estadias. Ela abre o arquivo para leitura ("rb"), lê cada estadia do arquivo usando fread() e imprime suas informações formatadas na tela.

h. void CheckIn()

A função CheckIn permite que o usuário realize o check-in de uma estadia. Primeiro, ela lista todas as estadias disponíveis, solicita o código da estadia desejada, verifica se a estadia existe e, em caso afirmativo, marca o quarto correspondente como ocupado.

i. **void CheckOut()**

A função CheckOut permite que o usuário realize o check-out de uma estadia. Ela lista todas as estadias disponíveis, solicita o código da estadia desejada, verifica se a estadia existe e, em caso afirmativo, desocupa o quarto, calcula e informa o preço a ser pago e os pontos de fidelidade acumulados pelo cliente.

j. **void ListaEstadiaPorCliente()**

A função ListaEstadiaPorCliente lista todas as estadias de um cliente específico. Ela solicita o código do cliente, percorre o arquivo de estadias para encontrar todas as estadias do cliente e exibe as informações dessas estadias. Além disso, calcula e exibe os pontos de fidelidade acumulados pelo cliente.

4. **FUNCIONARIO.C**

a. **void flush_in ()**

Esta função é responsável por limpar o buffer de entrada (stdin). Ela descarta todos os caracteres pendentes no buffer de entrada até encontrar um caractere de nova linha (\n) ou o final do arquivo (EOF).

b. **void Continue()**

A função Continue() exibe uma mensagem para o usuário indicando que ele deve digitar algo para continuar. Ela espera por um único caractere de entrada do usuário antes de retornar ao programa principal.

c. **void CriaSeNaoExisteFuncionario()**

Esta função verifica se o arquivo que armazena os dados dos funcionários existe. Se não existir, ela cria um arquivo novo e inicializa com dados fictícios de funcionários.

d. **void ListaFuncionario()**

Esta função lista todos os funcionários cadastrados no sistema, lendo os dados do arquivo onde os funcionários estão armazenados.

e. **int QuantosFuncionarios()**

Retorna o número total de funcionários cadastrados no sistema.

f. void CadastraFuncionario()

Permite o cadastro de um novo funcionário, coletando informações como nome, telefone, cargo, senha e salário. Os dados são então escritos no arquivo de funcionários.

g. void PesquisaFuncionarioPorCodigo()

Permite pesquisar um funcionário específico pelo seu código. Retorna os dados do funcionário se encontrado.

h. void PesquisaFuncionarioPorNome()

Permite pesquisar funcionários cujo nome contenha uma determinada sequência de caracteres. Retorna todos os funcionários cujos nomes correspondem ao critério de pesquisa.

i. int Login(int codigo, char *senha)

Realiza o login de um funcionário no sistema, verificando se o código e a senha fornecidos correspondem a um funcionário registrado.

j. void PesquisaCliente()

Essa função apresenta um menu de opções para realizar diferentes tipos de pesquisas relacionadas a clientes, como pesquisa por nome, código, listar todos os clientes e pesquisar reserva (estadia).

k. void PesquisaFuncionario()

Similar à função PesquisaCliente(), esta função apresenta um menu para realizar diferentes tipos de pesquisas relacionadas a funcionários, como pesquisa por nome, código e listar todos os funcionários.

l. void Cadastro()

Esta função apresenta um menu de opções para realizar diferentes tipos de cadastros, como cadastro de funcionário, cliente, estadia e quartos.

m. void PesquisaEstadia ()

A função PesquisaEstadia apresenta ao usuário um menu de opções para pesquisar estadias. O usuário pode optar por listar todas as estadias e reservas marcadas, pesquisar estadias por clientes, ou voltar ao menu anterior. Dependendo da escolha do usuário, a função chama outras funções para realizar a tarefa desejada.

n. void Pesquisa()

Apresenta um menu de opções para realizar pesquisas tanto de clientes quanto de funcionários, além de permitir retornar ao menu anterior.

o. void MenuGerente()

Esta função representa o menu principal do sistema para o perfil de gerente. Apresenta opções para pesquisa, cadastro, realizar reserva, check-in, check-out e sair do sistema.

p. void MenuRecepcionista ()

Apresenta um menu de opções específico para o perfil de recepcionista. Oferece opções para cadastro de clientes, realizar reserva (não implementado), check-in e check-out.

q. void MenuAuxiliar()

Mostra um menu simplificado para o perfil de auxiliar de limpeza, permitindo apenas a checagem de quartos disponíveis.

5. QUARTO.C

a. void CriaSeNaoExisteQuarto()

Verifica se o arquivo de quartos existe. Se não existir, cria um arquivo inicial com informações de seis quartos.

b. void ListaQuartos()

Lista todos os quartos presentes no arquivo room.dat.

c. int quantosQuartos()

Permite o cadastro de um novo quarto no arquivo room.dat.

d. bool CodigoJaExiste(int codigo)

Verificar se um código de quarto já existe no arquivo room.dat.

e. void CadastraQuarto()

Cadastrar um novo quarto no arquivo room.dat.

f. void OcuparQuarto(int codigo)

Marcar um quarto como ocupado no arquivo room.dat com base no número do quarto fornecido.

g. void DesocuparQuarto(int codigo)

Marcar um quarto como desocupado no arquivo room.dat com base no número do quarto fornecido.

h. bool QuartoExiste(int codigo)

A função QuartoExiste verifica se um quarto com o código especificado existe no arquivo de dados.

i. bool QuartoSuporta(int codigo, int hospedes)

A função QuartoSuporta verifica se um quarto com o código especificado pode acomodar a quantidade de hóspedes fornecida.

j. bool QuartoOcupado(int codigo)

A função QuartoOcupado verifica se um quarto com o código especificado está atualmente ocupado.