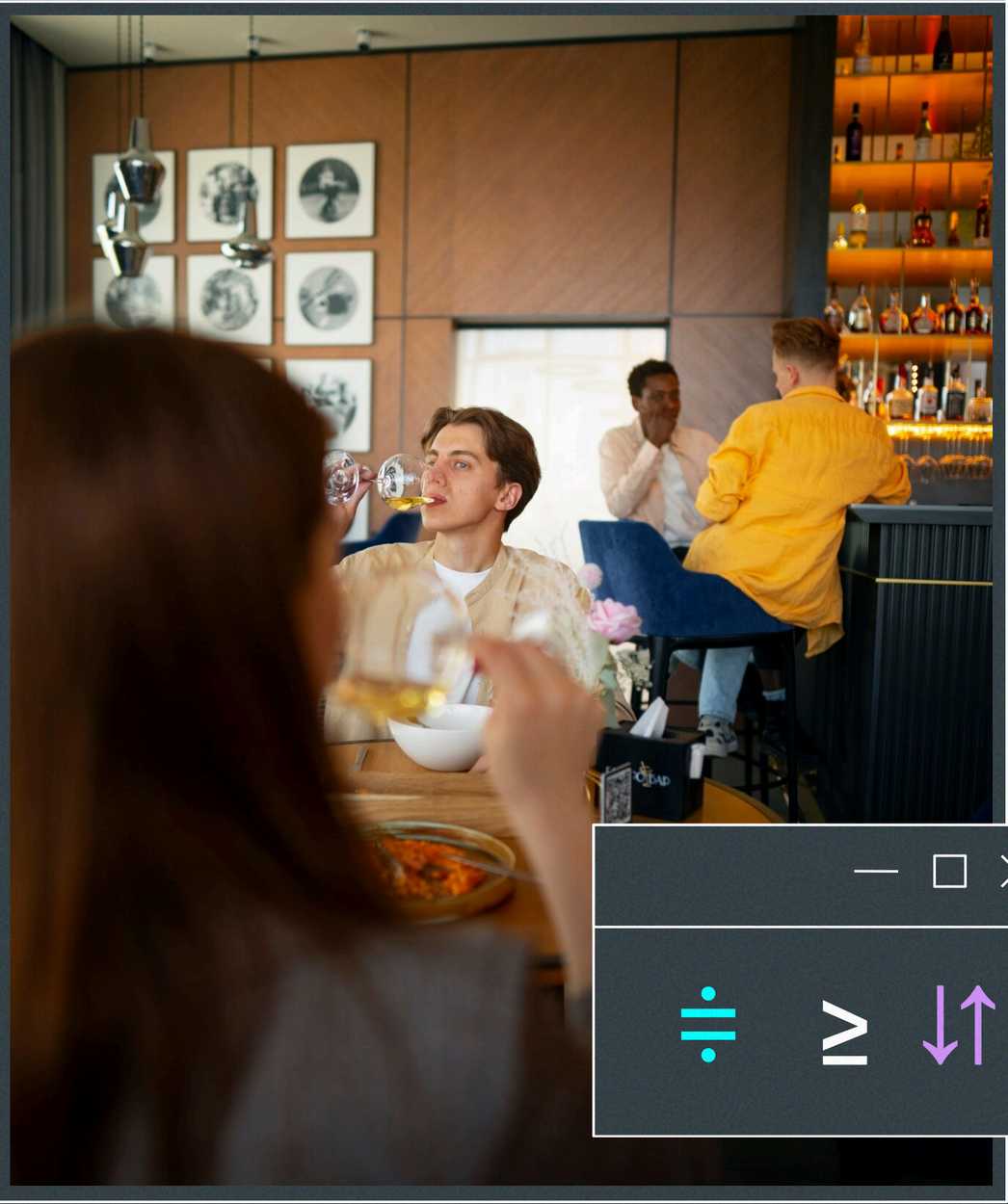
Gerenciamento de Reservas de Restaurante em Linguagem C

Felipe Lins
Camille
Thiago Gerbi
Rafael Galindo
Vitor Tamais
Murilo Bastos

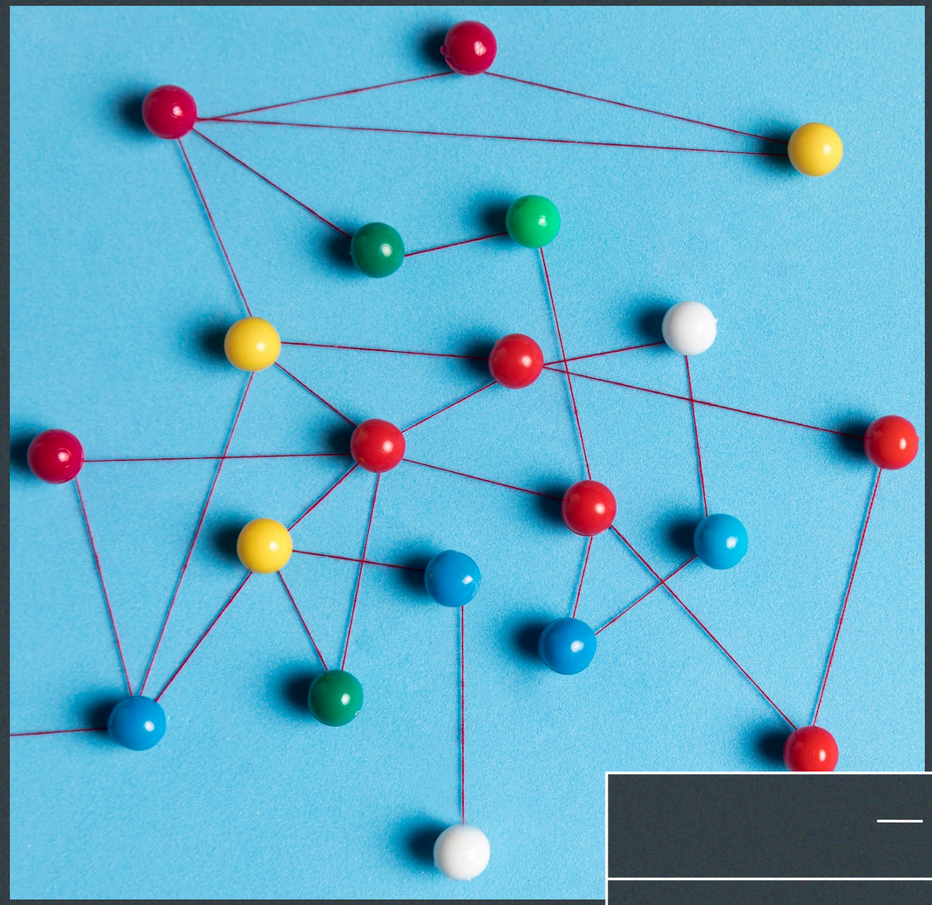
Contexto

Gerenciamento de Reservas em um Restaurante: O código implementa um sistema para gerenciar reservas de clientes em um restaurante. Ele permite adicionar, remover, exibir e consultar reservas, além de gerar relatórios de ocupação. As reservas incluem informações como nome, sobrenome, número de pessoas, data e horário.



Estruturas de Dados

As reservas são armazenadas em uma lista encadeada, onde cada nó contém uma reserva. Funções são fornecidas para manipular essa lista, como adicionar e remover reservas, exibir todas as reservas, consultar reservas específicas por data e horário, e gerar um relatório de ocupação para um intervalo de datas.



Abstração de Dados

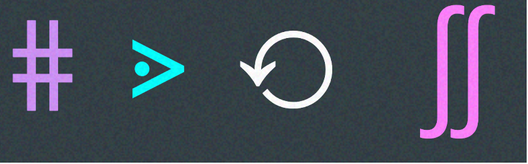
```c
typedef struct {
    char nome[50];
    char sobrenome[50];
    int num_pessoas;
    char data[20];
    char horario[10];
} Reserva;

typedef struct No {
    Reserva reserva;
    struct No *prox;
} No;
```

A abstração de dados fornecida no código é realizada através da definição de estruturas e funções que manipulam essas estruturas. A estruturaarmazena os detalhes de uma reserva (nome, sobrenome, número de pessoas, data e horário).

```c
No *criarNo(Reserva reserva) {
    No *novo = (No *)malloc(sizeof(No));  // Aloca memoria para um novo no
    if (novo == NULL) {
        printf("Erro ao alocar memória!\n");
        exit(1);
    }
    novo->reserva = reserva;
    novo->prox = NULL;  // Inicializa o ponteiro para o proximo no como NULL
    return novo;
}
```

Ponteiros

Os ponteiros permitem a criação e manipulação dinâmica da lista de reservas, onde cada nó aponta para o próximo, formando uma cadeia de nós.



Structs

Struct Reserva:

Armazena os detalhes de uma reserva (nome, sobrenome, número de pessoas, data e horário).

```c
typedef struct {
  char nome[50];
  char sobrenome[50];
  int num_pessoas;
  char data[20];
  char horario[10];
} Reserva;
```

Struct No:

Representa um nó na lista encadeada, contendo uma reserva e um ponteiro para o próximo nó.

```c
typedef struct No {
  Reserva reserva;
  struct No *prox;
} No;
```

Lista Encadeada de Reservas:

```c
void adicionarReserva(No **lista, Reserva reserva) {
    No *novo = criarNo(reserva);  // Cria um novo no com a reserva
    if (*lista == NULL) {
        *lista = novo;  // Adiciona o primeiro no na lista
    } else {
        No *atual = *lista;
        while (atual->prox != NULL) {
            atual = atual->prox;
        }
        atual->prox = novo;  // Adiciona o novo no ao final da lista
    }
}
```

O que faz:

Armazena as reservas do restaurante. Cada nó na lista contém uma reserva e um ponteiro para o próximo nó.

Como funciona:

Permite adicionar novas reservas ao final, remover reservas específicas e percorrer a lista para exibir ou consultar reservas.

Teste

```
Menu:
1. Adicionar Reserva
2. Remover Reserva
3. Exibir Reservas
4. Consultar Reservas em Horário Específico
5. Gerar Relatório de Ocupação
6. Sair
Escolha uma opção: 1

Insira os dados da reserva:
Nome do cliente: Thiago
Número de pessoas: 10
Data (AAAA-MM-DD): 2024-06-12
Horário (HH:MM): 20:00
```

1

```
Menu:
1. Adicionar Reserva
2. Remover Reserva
3. Exibir Reservas
4. Consultar Reservas em Horário Específico
5. Gerar Relatório de Ocupação
6. Sair
Escolha uma opção: 3
Lista de Reservas:
Nome: Thiago, Num. de Pessoas: 10, Data: 2024-06-12, Horário: 20:00
```

2

```
Menu:
1. Adicionar Reserva
2. Remover Reserva
3. Exibir Reservas
4. Consultar Reservas em Horário Específico
5. Gerar Relatório de Ocupação
6. Sair
Escolha uma opção: 2

Insira o nome do cliente para remover a reserva: Thiago
Reserva de Thiago removida com sucesso!

Menu:
1. Adicionar Reserva
2. Remover Reserva
3. Exibir Reservas
4. Consultar Reservas em Horário Específico
5. Gerar Relatório de Ocupação
6. Sair
Escolha uma opção: 3
Lista de Reservas:
```

3

```
2. Remover Reserva
3. Exibir Reservas
4. Consultar Reservas em Horário Específico
5. Gerar Relatório de Ocupação
6. Sair
Escolha uma opção: 1

Insira os dados da reserva:
Nome do cliente: Thiago
Número de pessoas: 5
Data (AAAA-MM-DD): 2024-06-12
Horário (HH:MM): 21:00

Menu:
1. Adicionar Reserva
2. Remover Reserva
3. Exibir Reservas
4. Consultar Reservas em Horário Específico
5. Gerar Relatório de Ocupação
6. Sair
Escolha uma opção: 3
Lista de Reservas:
Nome: Thiago, Num. de Pessoas: 14, Data: 2024-06-12, Horário: 19:00
Nome: Thiago, Num. de Pessoas: 5, Data: 2024-06-12, Horário: 21:00
```

4

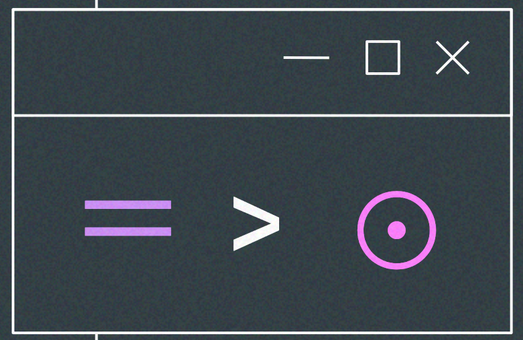
Teste

```
Menu:
1. Adicionar Reserva
2. Remover Reserva
3. Exibir Reservas
4. Consultar Reservas em Horário Específico
5. Gerar Relatório de Ocupação
6. Sair
Escolha uma opção: 1

Insira os dados da reserva:
Nome do cliente: Felipe
Sobrenome do cliente: Lins
Número de pessoas: 1
Data (DD-MM-AAAA): 24-07-2024
Horário (HH:MM): 19:30

Menu:
1. Adicionar Reserva
2. Remover Reserva
3. Exibir Reservas
4. Consultar Reservas em Horário Específico
5. Gerar Relatório de Ocupação
6. Sair
Escolha uma opção: 3
Lista de Reservas:
Nome: Felipe Lins, Num. de Pessoas: 1, Data: 24-07-2024, Horário: 19:30

Menu:
1. Adicionar Reserva
2. Remover Reserva
3. Exibir Reservas
4. Consultar Reservas em Horário Específico
5. Gerar Relatório de Ocupação
6. Sair
Escolha uma opção: 5

Intervalo de tempo para o relatório:
Data de início (DD-MM-AAAA): 10:15
Data inválida. Insira novamente (formato: DD-MM-AAAA): 15-08-2024
Data de fim (DD-MM-AAAA): 16-02-2025

Relatório de Ocupação do Restaurante:
Total de lugares ocupados: 0
```

```
Menu:
1. Adicionar Reserva
2. Remover Reserva
3. Exibir Reservas
4. Consultar Reservas em Horário Específico
5. Gerar Relatório de Ocupação
6. Sair
Escolha uma opção: 2

Insira o nome e sobrenome do cliente para remover a reserva: Felipe Lins
Reserva de Felipe Lins removida com sucesso!
```

Conclusão Geral

O código é um exemplo funcional de gerenciamento de reservas, demonstrando o uso de listas encadeadas para armazenar e manipular dados de maneira dinâmica. Este sistema é eficaz para pequenas e médias aplicações onde a gestão de reservas precisa ser flexível e adaptável, permitindo operações de inserção, remoção e consulta de forma eficiente.



Obrigado!