Centro Universitário de Excelência
Sistemas de Informação

Big Price

O título deveria ficar em centralizado. O valor vai ficar centralizado.

Autores: Eric Antunes
Danilo Alexandre
Valnei Sousa

O nome dos autores deveria ficar em ordem alfabética

Feira de Santana, 2025

Agenda

1. **Tela inicial e Menu principal**
2. **Adição de Itens**
3. **Atualização e Gerenciamento de itens**
4. **Criação de Pedido e Adição de Desconto**
5. **Atualização de Pedido**
6. **Consulta de Pedido**

Tela Inicial

Menu principal: PrintLn

```
// MENU PRINCIPAL
println("\nFOODDELIVERY ")
println("1. Cadastrar Item")
println("2. Atualizar Item")
println("3. Criar Pedido")
println("4. Atualizar Pedido")
println("5. Consultar Pedidos")
println("0. Sair")
print("Escolha uma opção: ")
```

```
FOODDELIVERY
1. Cadastrar Item
2. Atualizar Item
3. Criar Pedido
4. Atualizar Pedido
5. Consultar Pedidos
0. Sair
Escolha uma opção:
```

foi possível o uso dos prints para representar a experiência do usuário.

Aumentar o tamanho das imagens.

legal seria se fosse com o fundo branco

Consulta e Gerenciamento de itens

Adição de itens

```
when (readln()) {
    //CADASTRAR ITEM
    "1" -> {
        print("Nome do item: ")
        val nome = readln()
        print("Descrição do item: ")
        val descricao = readln()
        print("Preço (ex: 10.50): ")
        val preco = readln().replace(',', '.').toDoubleOrNull() ?: 0.0
        print("Quantidade em estoque: ")
        val quantidade = readln().toIntOrNull() ?: 0

        itens.add(
            mutableMapOf(
                "codigo" to codigoAtual,
                "nome" to nome,
                "descricao" to descricao,
                "preco" to preco,
                "quantidade" to quantidade
            )
        )
        println("Item cadastrado! Código: $codigoAtual")
        codigoAtual++
    }
```

```
Escolha uma opção: 1
Nome do item: Pizza de Calabresa
Descrição do item: Pizza de Calabresa com Frango
Preço (ex: 10.50): 10,50
Quantidade em estoque: 5
Item cadastrado! Código: 1
```

O ideal era ser com o fundo branco.

aqui teria sido legal explicar a escolha da estrutura de dados

O foco da experiência seria legal para explorar para a próx apresentação.

Big Price

Consulta e Gerenciamento de itens

Atualização/Busca de itens

```
//ATUALIZAR ITEM
"2" -> when {
    itens.isEmpty() -> println("Nenhum item cadastrado.")
    else -> {
        println("Itens cadastrados:")
        itens.forEach { item ->
            println("Código ${item["codigo"]} - ${item["nome"]}: ${item["descricao"]} " +
                    "(R$ ${"%.2f".format( ...args = item["preco"])}, ${item["quantidade"]} unid)")
        }
        print("Digite o código do item para atualizar: ")
        val cod = readln().toIntOrNull()
        val item = itens.find { it["codigo"] == cod }
        when {
            item == null -> println("Código inválido!")
            else -> {
                print("Novo nome (${item["nome"]}): ")
                val n = readln()
                if (n.isNotBlank()) item["nome"] = n

                print("Nova descrição (${item["descricao"]}): ")
                val d = readln()
                if (d.isNotBlank()) item["descricao"] = d

                print("Novo preço (${item["preco"]}): ")
                val p = readln()
                if (p.isNotBlank()) item["preco"] = p.replace( oldChar = ',', newChar = '.').toDouble()

                print("Nova quantidade (${item["quantidade"]}): ")
                val q = readln()
                if (q.isNotBlank()) item["quantidade"] = q.toInt()

                println("Item atualizado!")
            }
        }
```

o find sempre retorna apenas um valor.

a validação era importante

```
Escolha uma opção: 2
Itens cadastrados:
Código 1 - hambúrguer: carne,cebola, picles, ketchup, mostarda e pão sem gergelim. (R$ 12,50, 20 unid)
Digite o código do item para atualizar: 1
Novo nome (hambúrguer): burguer
Nova descrição (carne,cebola, picles, ketchup, mostarda e pão sem gergelim.): carne,cebola, picles, ketchup
Novo preço (12.5): 10.99
Nova quantidade (20): 32
Item atualizado!
```

imagem comprimida.

Price

Consulta e Gerenciamento de itens

Criação de Pedido e Adição de Desconto

```
//CRIAR PEDIDO
"3" -> when {
    itens.isEmpty() -> println("Nenhum item disponivel.")
    else -> {
        println("Itens disponíveis:")
        itens.forEach { item ->
            println("Código ${item["codigo"]} - ${item["nome"]}: ${item["descricao"]} " +
                    "(R$ ${"%.2f".format( ...args = item["preco"])}, ${item["quantidade"]} unid)")
        }
        print("Digite o código do item para pedir: ")
        val cod = readln().toIntOrNull()
        val item = itens.find { it["codigo"] == cod }

        when {
            item == null || (item["quantidade"] as Int) <= 0 -> println("Código inválido ou sem estoque!")
            else -> {
                print("Quantidade: ")
                val qtd = readln().toIntOrNull() ?: 0
                when {
                    qtd <= 0 -> println("Quantidade inválida!")
                    qtd > (item["quantidade"] as Int) -> println("Estoque insuficiente!")
                    else -> {
                        print("Possui cupom de desconto? (S/N): ")
                        val cupom = readln()
                        val desconto = if (cupom.equals( other = "S",  ignoreCase = true)) 0.1 else 0.0 // 10% desconto

                        val precoTotal = (item["preco"] as Double) * qtd
                        val precoFinal = precoTotal - (precoTotal * desconto)

                        pedidos.add(
                            mutableMapOf(
                                "item" to item,
                                "quantidade" to qtd,
                                "status" to "ACEITO",
                                "precoFinal" to precoFinal
                            )
                        )
                        item["quantidade"] = (item["quantidade"] as Int) - qtd
                        println("Pedido criado! Valor final: R$ ${"%.2f".format( ...args = precoFinal)}")
                    }
```

```
Escolha uma opção: 3
Itens disponíveis:
Código 1 - burguer: carne,cebola, picles, ketchup (R$ 10,99, 32 unid)
Digite o código do item para pedir: 1
Quantidade: 3
Possui cupom de desconto? (S/N): n
Pedido criado! Valor final: R$ 32,97
```

vocês deveriam apresentar sobre a estrutura de dados que usaram List<Map<String, Any>> para representar a lista de pedidos

Big Price

Consulta e Gerenciamento de itens

Atualização de Pedido

```
//ATUALIZAR PEDIDO
"4" -> when {
    pedidos.isEmpty() -> println("Nenhum pedido registrado.")
    else -> {
        println("Pedidos:")
        pedidos.forEachIndexed { index, pedido ->
            println("${index + 1} - ${(pedido["item"] as MutableMap<String, Any>)["nome"]} - " +
                    "${pedido["quantidade"]} unid (Status: ${pedido["status"]})")
        }
        print("Número do pedido para atualizar: ")
        val i = readln().toIntOrNull()?.minus( other = 1)
        when {
            i == null || i !in pedidos.indices -> println("Pedido não encontrado!")
            else -> {
                println("Escolha novo status: ")
                println("1 - FAZENDO\n2 - FEITO\n3 - ESPERANDO ENTREGADOR\n4 - SAIU PARA ENTREGA\n5 - ENTREGUE")
                when (readln()) {
                    "1" -> pedidos[i]["status"] = "FAZENDO"
                    "2" -> pedidos[i]["status"] = "FEITO"
                    "3" -> pedidos[i]["status"] = "ESPERANDO ENTREGADOR"
                    "4" -> pedidos[i]["status"] = "SAIU PARA ENTREGA"
                    "5" -> pedidos[i]["status"] = "ENTREGUE"
                    else -> println("Status inválido!")
                }
                println("Status atualizado!")
            }
        }
    }
}
```

Nesta atualização, precisamos melhorar o código.

```
Escolha uma opção: 4
Pedidos:
1 - burguer - 3 unid (Status: ACEITO)
Número do pedido para atualizar: 1
Escolha novo status:
1 - FAZENDO
2 - FEITO
3 - ESPERANDO ENTREGADOR
4 - SAIU PARA ENTREGA
5 - ENTREGUE
1
Status atualizado!
```

Consulta e Gerenciamento de itens

Consulta de Pedido

```
// CONSULTAR PEDIDOS
"5" -> when {
    pedidos.isEmpty() -> println("Nenhum pedido registrado.")
    else -> {
        println("CONSULTAR PEDIDOS")
        println("1 - Todos\n2 - ACEITO\n3 - FAZENDO\n4 - FEITO\n5 - ESPERANDO ENTREGADOR\n6 - SAIU PARA ENTREGA\n7 - ENTREGUE")
        print("Escolha uma opção: ")
        val opc = readln()
        pedidos.forEachIndexed { index, pedido ->
            val status = pedido["status"]
            when (opc) {
                "1" -> println("${index + 1} - ${(pedido["item"] as MutableMap<String, Any>)["nome"]} " +
                        "(${pedido["quantidade"]} unid) - Status: $status")
                "2" -> if (status == "ACEITO") println("${index + 1} - ${(pedido["item"]
                        as MutableMap<String, Any>)["nome"]} - $status")
                "3" -> if (status == "FAZENDO") println("${index + 1} - ${(pedido["item"]
                        as MutableMap<String, Any>)["nome"]} - $status")
                "4" -> if (status == "FEITO") println("${index + 1} - ${(pedido["item"]
                        as MutableMap<String, Any>)["nome"]} - $status")
                "5" -> if (status == "ESPERANDO ENTREGADOR") println("${index + 1} - " +
                        "${(pedido["item"] as MutableMap<String, Any>)["nome"]} - $status")
                "6" -> if (status == "SAIU PARA ENTREGA") println("${index + 1} - " +
                        "${(pedido["item"] as MutableMap<String, Any>)["nome"]} - $status")
                "7" -> if (status == "ENTREGUE") println("${index + 1} - " +
                        "${(pedido["item"] as MutableMap<String, Any>)["nome"]} - $status")
            }
        }
    }
}
```

```
Escolha uma opção: 5
CONSULTAR PEDIDOS
1 - Todos
2 - ACEITO
3 - FAZENDO
4 - FEITO
5 - ESPERANDO ENTREGADOR
6 - SAIU PARA ENTREGA
7 - ENTREGUE
Escolha uma opção: 1
1 - burguer (3 unid) - Status: FAZENDO
```

Era melhor demonstrar a experiência do usuário.

- Os slides apresentaram alguns problemas de formatação.
- As imagens poderiam representar melhor a experiência do usuário.
- Melhorar a qualidade das imagens.

→ O tempo da apresentação foi OK.

→ Valnei chegou atrasado e por isso vai ficar com 30% da nota.

Sobre as questões técnicas temos alguns pontos:

do-While → a explicação sobre o uso do do..While não satisfatória, apesar da explicação de Danilo.

problema do Zero - valores atribuídos com padrão zero foi bem explicado.

itens no pedido - a explicação sobre um único item no pedido não foi satisfatória, nem mesmo a explicação.

A nota da apresentação ficou:

AO → 1.5 pontos