Módulo UUID

O módulo **uuid** fornece funções para geração de identificadores únicos universais (UUIDs) seguindo o padrão RFC 4122. Este módulo é útil para scripts que precisam criar identificadores únicos para rastreamento de transações, identificação de recursos, e casos que requerem unicidade garantida.

Características principais:

- Geração de UUID versão 4 (aleatório)
- Formato padrão RFC 4122 (8-4-4-4-12)
- Garantia de unicidade estatística
- Thread-safe e seguro para uso concorrente

Formato UUID v4:

- `xxxxxxxx-xxxx-4xxx-yxxx-xxxxxxxxxxxx`
- Onde `x` é qualquer dígito hexadecimal
- `4` indica versão 4 (aleatório)
- `y` é 8, 9, A ou B (indicando variante)

Funções Disponíveis

1. `uuid.uuid4()`

Gera um UUID versão 4 (aleatório) no formato padrão RFC 4122.

Parâmetros:

- Nenhum

Retorno:

- **string**: UUID no formato `xxxxxxxx-xxxx-4xxx-yxxx-xxxxxxxxxxxx`

Comportamento:

1. Gera 128 bits aleatórios usando gerador criptograficamente seguro
2. Define os bits de versão para 4 (0100)

3. Define os bits de variante para RFC 4122 (10)
4. Formata como string no padrão 8-4-4-4-12
5. Garante unicidade estatística (probabilidade de colisão extremamente baixa)

Exemplo de Uso:

```
-- Gerar um UUID simples
local id = uuid.uuid4()
print("UUID gerado:", id)
-- Exemplo: "f47ac10b-58cc-4372-a567-0e02b2c3d479"

-- Gerar múltiplos UUIDs
local ids = {}
for i = 1, 5 do
    ids[i] = uuid.uuid4()
    print("UUID", i, ":", ids[i])
end

-- Usar como identificador de transação
local transacao_id = uuid.uuid4()
local log_entry = {
    id = transacao_id,
    tipo = "pagamento",
    valor = 150.75,
    timestamp = now(),
    status = "processando"
}
print("Transação ID:", transacao_id)

-- Criar nomes de arquivo únicos
local nome_arquivo = "backup_" .. uuid.uuid4() .. ".tar.gz"
print("Arquivo de backup:", nome_arquivo)

-- Gerar token de sessão
local sessao_token = "sess_" .. string.sub(uuid.uuid4(), 1, 8)
print("Token de sessão:", sessao_token)
```

Informações Adicionais

Unicidade Garantida:

```
-- Probabilidade de colisão extremamente baixa
-- 2^128 possibilidades ≈ 3.4 × 10^38 UUIDs únicos
-- Mesmo gerando 1 bilhão de UUIDs por segundo,
-- levaria ~100 anos para ter 50% de chance de colisão

local function testar_unicidade(iteracoes)
    local uuids = {}
    local colisoes = 0

    for i = 1, iteracoes do
        local id = uuid.uuid4()

        if uuids[id] then
            colisoes = colisoes + 1
            log.error("COLISÃO DETECTADA!", "Iteração:", i, "ID:", id)
        else
            uuids[id] = true
        end
    end

    return {
        iteracoes = iteracoes,
        colisoes = colisoes,
        taxa_colisao = (colisoes / iteracoes) * 100
    }
end

-- Em testes práticos, colisões são virtualmente inexistentes
```

Formato Padrão:

```
-- UUID sempre no formato RFC 4122
local id = uuid.uuid4()
-- Formato: xxxxxxxx-xxxx-4xxx-yxxx-xxxxxxxxxxxx
-- Exemplo: "f47ac10b-58cc-4372-a567-0e02b2c3d479"
```

```
-- Validar formato
local function validar_uuid(uuid_str)
    local padrao = "^%x%x%x%x%x%x%x%x-%x%x%x%x-4%x%x%x-[89ab]%x%x%x-
%x%x%x%x%x%x%x%x%x%x%x%x$"
    return string.match(uuid_str:lower(), padrao) ~= nil
end

print("UUID válido?", validar_uuid(id))  -- true
print("UUID válido?", validar_uuid("invalid"))  -- false
```

Exemplos de Uso

1. Nomes de Arquivos Temporários:

```
-- Gerar nomes de arquivos únicos para processamento
local function criar_arquivo_temporario(extensao, dados)
    local nome_arquivo = "tmp_" .. uuid.uuid4() .. "." .. (extensao or "tmp")
    local caminho = "/tmp/" .. nome_arquivo

    -- Escrever dados
    local arquivo = io.open(caminho, "w")
    if arquivo then
        arquivo:write(dados)
        arquivo:close()

        -- Registrar para limpeza
        local arquivos_temp, _ = store.get("arquivos_temporarios") or {}
        table.insert(arquivos_temp, {
            caminho = caminho,
            criado_em = now(),
            expira_em = now() + 3600  -- 1 hora
        })
        store.put("arquivos_temporarios", arquivos_temp)

        return caminho
    else
        return nil, "Não foi possível criar arquivo"
```

```
    end
end

-- Limpar arquivos temporários antigos
local function limpar_arquivos_temporarios()
    local arquivos_temp, _ = store.get("arquivos_temporarios") or {}
    local removidos = 0
    local agora = now()

    for i = #arquivos_temp, 1, -1 do
        local arquivo = arquivos_temp[i]

        if agora > arquivo.expira_em then
            -- Tentar remover arquivo
            local ok, _ = pcall(os.remove, arquivo.caminho)
            if ok then
                table.remove(arquivos_temp, i)
                removidos = removidos + 1
                log.debug("Arquivo temporário removido", "Caminho:",
arquivo.caminho)
            end
        end
    end

    store.put("arquivos_temporarios", arquivos_temp)
    return removidos
end
```

Revision #1
Created 3 February 2026 16:47:39 by Marc
Updated 3 February 2026 16:47:50 by Marc