

# Especificação do Sistema de Sugestão de Apostas para Mega-Sena

## 1. Introdução

Este documento descreve a especificação de um sistema em Java para geração de sugestões de apostas para a Mega-Sena, utilizando estatísticas do banco de dados SQLite disponível em <https://megapower-loterias.com.br/dados/MegaPower.db>. O sistema será multithread, utilizará um pool de conexões com SQLite, e implementará estratégias de ranking para selecionar as melhores apostas com base em análises estatísticas. A saída será formatada com mensagens NLS-compatíveis (Português do Brasil como padrão) e exibirá data, hora e valores conforme o ambiente operacional.

---

## 2. Requisitos Funcionais

### <sup>1</sup> Geração de Apostas:

- O sistema deve gerar um número configurável de apostas para o próximo concurso da Mega-Sena, informado via parâmetros POSIX.
- As apostas podem ser geradas de três formas:
  - **Aleatória:** N apostas geradas aleatoriamente.
  - **Índice Lexicográfico:** N apostas baseadas em índices lexicográficos entre 1 ([01-02-03-04-05-06]) e 50.063.860 ([55-56-57-58-59-60]).
  - **Avaliação Completa:** Todas as 50.063.860 combinações possíveis avaliadas para selecionar as N melhores.
- Cada aposta consiste em 6 dezenas (valores de 1 a 60).

## <sup>2</sup> Estratégias de Ranking:

- As apostas serão avaliadas por uma cadeia de estratégias implementadas como classes derivadas de uma classe abstrata ou interface.
- Estratégias incluem, mas não se limitam a:
  - **Ranking Gaussiano:** Baseado em múltiplos critérios estatísticos.
  - **Análise de Transição de Estados:** Usando a tabela `EST_MUD_ESTADO`.
  - **Probabilidades Condicionais:** Por posição, com base no histórico (`AGREG_DIS_DEZ_POSICAO`).
  - **Sistema de Scoring com Pesos:** Ponderação estatística para critérios combinados.
  - Outras estratégias podem ser implementadas conforme análise do desenvolvedor.
- O ranking final de uma aposta é a soma dos scores atribuídos por todas as estratégias.

## <sup>3</sup> Exibição de Resultados:

- Mensagens de saída devem respeitar NLS do ambiente operacional, com fallback para Português do Brasil.
- As mensagens incluirão data, hora e valores formatados adequadamente.

## <sup>4</sup> Análise de Desempenho:

- Comparar sugestões de apostas armazenadas na tabela `FAT_APOSTAS` com o resultado do último sorteio.
- A análise incluirá data da aposta, data do sorteio, número do concurso, soma do ranking das estratégias e comparação com o resultado real.

---

## 3. Requisitos Não Funcionais

## <sup>1</sup> Multithreading:

- Utilizar `ExecutorService` para gerenciar threads.
- Processamento de cálculos estatísticos distribuído em múltiplos núcleos.
- Uso de `Stream` paralelos para eficiência no processamento de grandes conjuntos de dados.

## <sup>2</sup> Gerenciamento de Conexões com Banco de Dados:

- Implementar pool de conexões com Apache Commons DBCP.
- Configurar timeout de 5 segundos para conexões.
- Limite de conexões ativas para evitar "host busy".
- Fechamento adequado de conexões com `try-with-resources`.

## <sup>3</sup> Resiliência:

- **Sistema de Retry:** Até 3 tentativas para falhas temporárias.
- **Backoff Exponencial:** Intervalos crescentes entre tentativas (e.g., 100ms, 200ms, 400ms).
- **Failover:** Após 3 falhas, o sistema deve logar o erro e continuar com a próxima tarefa.
- **Cache Inteligente:** Armazenar scores calculados localmente com validade de 1 minuto.

## <sup>4</sup> Gerenciamento de Recursos:

- Fechamento gracioso de threads com `ExecutorService.awaitTermination`.
- Tratamento robusto de exceções para evitar falhas silenciosas.

## <sup>5</sup> Desempenho:

- Otimizar consultas ao banco SQLite para minimizar gargalos.
- Usar índices nas tabelas do banco para consultas frequentes.

---

## 4. Arquitetura do Sistema

#### **4.1. Estrutura do Banco de Dados**

O banco de dados SQLite ( MegaPower.db ) contém as seguintes tabelas/views relevantes:

- **DEZ\_OCORRENCIAS:**

- Descrição: Contém informações granulares sobre a ocorrência de cada dezena em todos os sorteios.
- Colunas (exemplo): concurso , data , dezena , ocorrencia .
- Uso: Análise de frequência de dezenas.

- **AGREG\_DIS\_DEZ\_POSICAO:**

- Descrição: Agrega a quantidade de ocorrências de cada dezena por posição no sorteio (1<sup>a</sup> a 6<sup>a</sup> dezena).
- Colunas (exemplo): dezena , posicao , quantidade .
- Uso: Cálculo de probabilidades condicionais por posição.

- **EST\_MUD\_ESTADO:**

- Descrição: Registra transições de estados entre sorteios (e.g., mudanças nas dezenas sorteadas).
- Colunas (exemplo): concurso , dezena\_anterior , dezena\_atual , transicao .
- Uso: Análise de padrões de transição.

- **FAT\_APOSTAS:**

- Descrição: Armazena apostas sugeridas, com informações de ranking e resultados.
- Colunas (exemplo): id\_aposta , data\_aposta , data\_sorteio , concurso , dezenas , ranking\_total , resultado\_real .
- Uso: Comparação de desempenho das apostas sugeridas.

- **Outras Tabelas/Views (a documentar):**

- O banco pode conter tabelas adicionais com estatísticas agregadas ou granulares (e.g., frequência de pares, trios, etc.).
- Recomenda-se análise exploratória para identificar outras tabelas/views úteis.

## 4.2. Estrutura do Código

O sistema será modular, com os seguintes pacotes:

- **br.com.megasena.model:**
  - Classes para representar apostas, sorteios e estatísticas.
  - Exemplo: Aposta , Sorteio , EstatisticaDezena .
- **br.com.megasena.strategy:**
  - Interface BetStrategy com método calculateScore(Aposta) para calcular o score de uma aposta.
  - Implementações: GaussianRankingStrategy , StateTransitionStrategy , ConditionalProbabilityStrategy , WeightedScoringStrategy .
- **br.com.megasena.database:**
  - Classes para acesso ao banco SQLite.
  - DatabaseConnectionPool : Configuração do pool de conexões com Apache Commons DBCP.
  - DAOs: DezenaDAO , SorteioDAO , ApostadoDAO .
- **br.com.megasena.generator:**
  - Classes para geração de apostas.
  - RandomBetGenerator , LexicographicBetGenerator , FullCombinationGenerator .
- **br.com.megasena.evaluation:**
  - BetEvaluator : Coordena a avaliação de apostas usando a cadeia de estratégias.
  - ResultAnalyzer : Compara apostas sugeridas com resultados reais.
- **br.com.megasena.util:**
  - Utilitários para formatação NLS, logging, e gerenciamento de threads.

#### 4.3. Fluxo de Execução

## **1 Leitura de Parâmetros POSIX:**

- Usar biblioteca como `JCommander` para parsear argumentos (e.g., `--num-bets` , `--mode=random|lexicographic|full` ).

## **2 Conexão com Banco de Dados:**

- Inicializar pool de conexões com Apache Commons DBCP.

## **3 Geração de Apostas:**

- Dependendo do modo (`random` , `lexicographic` , `full`), gerar N apostas ou todas as combinações.

## **4 Avaliação de Apostas:**

- Submeter apostas à cadeia de estratégias em paralelo usando `ExecutorService` e `Stream` paralelos.
- Somar scores para obter ranking final.

## **5 Persistência:**

- Armazenar apostas sugeridas em `FAT_APOSTAS` .

## **6 Análise de Desempenho:**

- Comparar apostas de `FAT_APOSTAS` com o último sorteio.

## **7 Exibição de Resultados:**

- Formatar saída com data, hora e valores usando `java.text` (NLS).

---

## **5. Estratégias de Avaliação**

## **1 Ranking Gaussiano:**

- Usa distribuição normal para avaliar a probabilidade de dezenas com base em sua frequência histórica ( DEZ\_OCORRENCIAS ).
- Fórmula: Score =  $\sum (z\text{-score(dezena}_i) / 6)$ , onde z-score é calculado com média e desvio padrão das ocorrências.

## **2 Análise de Transição de Estados:**

- Usa EST\_MUD\_ESTADO para calcular probabilidades de transição entre dezenas sorteadas.
- Score baseado na probabilidade de uma aposta seguir padrões históricos de transição.

## **3 Probabilidades Condicionais por Posição:**

- Usa AGREG\_DIS\_DEZ\_POSICAO para calcular a probabilidade de uma dezena aparecer em uma posição específica.
- Score =  $\sum P(\text{dezena}_i | \text{posição}_i)$ .

## **4 Sistema de Scoring com Pesos:**

- Combina múltiplos critérios (frequência, posição, transições) com pesos configuráveis.
- Exemplo: Score = w1 \* ScoreGaussiano + w2 \* ScoreTransicao + w3 \* ScoreCondisional.

## **5 Estratégias Adicionais (sugestão):**

- **Análise de Pares/Trios:** Identificar combinações de 2 ou 3 dezenas frequentes.
- **Entropia de Apostas:** Priorizar apostas com maior diversidade estatística.
- **Tendências Temporais:** Considerar sorteios recentes para capturar padrões de curto prazo.

---

# **6. Tecnologias Utilizadas**

- **Linguagem:** Java 17 (LTS).
  - **Bibliotecas:**
    - SQLite JDBC: Conexão com banco SQLite.
    - Apache Commons DBCP: Pool de conexões.
    - JCommander: Parsing de parâmetros POSIX.
    - SLF4J: Logging.
    - Java Streams: Processamento paralelo.
  - **Ferramentas:**
    - Maven: Gerenciamento de dependências.
    - JUnit: Testes unitários.
- 

## 7. Configurações

- **Pool de Conexões:**

- Máximo de 10 conexões ativas.
- Timeout: 5 segundos.
- Validação de conexão: `SELECT 1`.

- **Cache:**

- Validade: 1 minuto.
- Implementação: `ConcurrentHashMap` com expiração baseada em timestamp.

- **Retry:**

- Máximo de 3 tentativas.
- Backoff: 100ms, 200ms, 400ms.

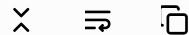
- **Threads:**

- Pool de threads configurado com  
`Executors.newFixedThreadPool(Runtime.getRuntime().availableProcessors())`.
  - `awaitTermination`: 60 segundos para desligamento.
- 

## 8. Saída do Sistema

- **Formato:** Texto formatado com `java.text` (NLS).
- **Exemplo:**

```
text
```



```
Data: 10/06/2025 13:19
Concurso: [NÚMERO]
Apostas Sugeridas:
1. [01-02-03-04-05-06] (Ranking: 85.5)
2. [07-08-09-10-11-12] (Ranking: 84.2)
...
Análise do Último Sorteio:
Concurso: [NÚMERO_ANTERIOR]
Resultado Real: [01-03-05-07-09-11]
Acertos da Sugestão: 4/6 (Ranking da Sugestão: 83.7)
```

## 9. Tratamento de Erros

- **Exceções SQL:** Capturadas e tratadas com retry/backoff.
- **Timeout de Conexão:** Failover após 3 tentativas.
- **Erros de Parsing POSIX:** Exibir mensagem de uso e encerrar.
- **Falhas de Thread:** Logar e continuar processamento com threads restantes.
- **Logging:** Usar SLF4J para registrar erros e eventos críticos.

## 10. Próximos Passos

- <sup>1</sup> Implementar classes base e estratégias iniciais.
- <sup>2</sup> Configurar pool de conexões e testes de acesso ao banco.
- <sup>3</sup> Desenvolver geradores de apostas (aleatório, lexicográfico, completo).
- <sup>4</sup> Testar desempenho com subconjunto de combinações.
- <sup>5</sup> Validar estratégias com sorteios históricos.

Mostrar na barra lateral

