



TenisLang

Linguagem de programação temática de tênis

Luise Pessoa Bastos





TenisLang é uma linguagem de domínio específico projetada para descrever e simular partidas de tênis de forma intuitiva e divertida. Seu programa começa com “warmup” e termina em “matchPoint”, delimitando claramente o escopo de uma partida. Entre esses marcadores, comandos como player, hits, ace, challenge e strategy permitem modelar pontos, saques especiais, replays de desafio e táticas reutilizáveis, tudo com sintaxe simples.



Cada instrução executada cumpre três etapas: o parser reconhece a estrutura, o codegen grava uma linha correspondente no arquivo out.ll (como um IR textual) e o runtime dispara imediatamente uma função que imprime a ação “ao vivo” no console, usando emojis e textos descritivos. Além disso, TennisLang suporta expressões aritméticas, booleanas (and, or) e relacionais (equal, greater, less), possibilitando condicionais e blocos de rally, letShot e tiebreak que espelham regras reais do esporte.



Criada para aproximar fãs de tênis ao universo de compiladores e design de linguagens, TennisLang une “jogo” e “código” de maneira lúdica, oferecendo feedback imediato e um histórico gravado em IR. É uma ferramenta excelente para quem quer aprender conceitos de parsing, geração de código e execução interativa — tudo dentro do contexto familiar de uma partida de tênis.

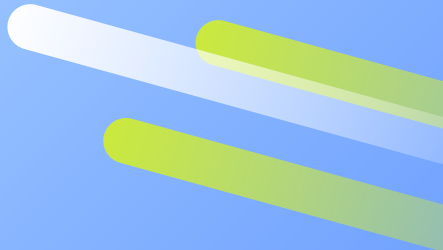



Motivação

- Ao usar um tema que muitos conhecem e gostam (o tênis), tornamos o aprendizado de conceitos de compilador, parsing e design de linguagens mais atrativo para os fãs do esporte
- Inspirar os amantes de tênis a programar
- Unir "jogos" e "código" de forma lúdica
- Relacionar estruturas de código (loops, condições) a situações reais de uma partida
- Fornecer feedback imediato com execução "ao vivo" a cada comando



Características

- **Warmup...matchPoint como delimitadores.** Essas duas palavras-chave marcam claramente o início e o fim de um script de TennisLang. Tudo que estiver entre *warmup* e *matchpoint* será interpretado como parte de partida. Isso facilita o parsing e delimita o escopo de comandos, assim como em um set de tênis onde o aquecimento prepara o jogo e o match point encerra o duelo.
 - **Comandos:** *player*, *hits*, *ace*, *strategy*, *tiebreak*, etc. `Challenge(<cond>)` serve ... `smash [replay ... smash]` - simula o pedido de revisão, com opção de replay.
 - **Geração de "out.ll" + execução "ao vivo":** Durante o parsing, cada comando `emit_*` grava uma linha no arquivo *out.ll*, criando um IR textual que reflete passo a passo da partida. Simultaneamente, as mesmas ações disparam funções de runtime (`shout()`, `play_move()`, `ace()`, etc.), exibindo no console a partida “ao vivo” com emojis e descrições legíveis. Essa dualidade permite tanto análise posterior do IR (para debug ou visualização) quanto feedback imediato para o usuário, tornando o desenvolvimento interativo e divertido.
- 
- 

Curiosidades



**crowd()
net()**

As chamadas built-in `crowd()` e `net()` não são meros tokens vazios: elas simulam o comportamento da plateia e o estado da rede. Você pode usar ``crowd() > 5000`` para decidir, por exemplo, se a pressão do público influencia um ``challenge``, ou ``if net()`` para tratar situações em que a bola toca na rede.



**letShot
challenge**

-`letShot(cond)` serve ... *smash` recria a regra do “let” — quando o saque toca na rede e deve ser repetido.* ``challenge(cond)` serve ... *smash [replay ... smash]` permite modelar o pedido de revisão do placar, incluindo replay opcional: exatamente como no tênis real.*



**Serve como
Bloco de Rally**

- O keyword ``serve`` inicia um `_rallyBlock_` e ``smash`` o encerra, organizando sequências de jogadas de maneira clara, assim como o formato real de um rally.

Exemplos de Código

Exemplo #1

```
✓ warmup  
  player federer is point;  
  federer hits 15;  
  williams hits 30;  
matchPoint
```

1. **Delimitadores do programa:**
 - "Warmup" inicia a seção de comandos de partida
 - "matchPoint" encerra o script
 - Tudo que vier entre essas palavras será processado pela TennisLang
2. **Declaração do jogador:**
 - Cria um jogador com identificador "federer" e tipo point. Registra o jogador federer como participante que pontua
3. **Comandos de jogada:**
 - Faz com que o jogador federer registre 15 pontos
 - Faz com que o jogador williams registre 30 pontos

Exemplos de Código

Exemplo #2

```
warmup
  player federer is point;
  challenge(williams equal 30) serve
    williams hits 40;
  smash
matchPoint
```

1. **Delimitadores do programa:**
 - "Warmup" inicia a seção de comandos de partida
 - "matchPoint" encerra o script
 - Tudo que vier entre essas palavras será processado pela TennisLang
2. **Declaração do jogador:**
 - Cria um jogador com identificador "federer" e tipo point, registrando-o como participante que pontua
3. **Desafio (challenge):**
 - Inicia um rally de desafio se a pontuação de williams for igual a 30
 - Dentro desse rally, williams marca 40 pontos
 - Smash encerra o rally de desafio



Regras de Execução

O interpretador só começa a processar o script quando encontra a palavra `warmup`, momento em que inicializa o gerador de IR (`out.ll`) e o runtime “ao vivo”. A partir daí, cada comando entre `warmup` e `matchPoint` é executado em três fases:

1. **Parsing:** o Flex/Bison reconhece a estrutura sintática do comando.
2. **Geração de IR:** é emitida uma linha correspondente no arquivo `out.ll`.
3. **Execução imediata:** a função de runtime associada (por exemplo, `shout()`, `play_move()`, `ace()`) é chamada e imprime no console.

Blocos especiais – rally (serve ... smash), challenge, letShot e tiebreak – só são executados se a condição relacional que os precede for verdadeira. Se um challenge incluir a palavra `replay`, esse segundo bloco também será processado da mesma forma. Comandos simples, como `<id> hits <expr>;` ou `shout(<expr>;` seguem exatamente a mesma sequência `parse` → `codegen` → `runtime`.

Quando o interpretador encontra `matchPoint`, ele finaliza o arquivo `out.ll` escrevendo um rodapé e encerra o runtime. Qualquer erro de sintaxe ou uso inválido de um comando interrompe imediatamente o processo, imprime uma mensagem de erro em `stderr` e não gera mais linhas de IR. Dessa forma, TennisLang garante que cada jogada seja registrada e exibida de maneira previsível e consistente.

