

# Architektur und Projektuebersicht

TinyPI0 ist in drei Projekte gegliedert:

- PI0.CLI: Kommandozeilenwerkzeug fuer Kompilierung und Ausfuehrung.
- PI0.Core: Lexer, Parser, Compiler und P-Code-Serialisierung.
- PI0.Vm: P-Code-Interpreter inkl. I/O-Adapter.

Die detaillierte Architektur sowie weitere Referenzen werden im Kapitel "Inhalte aus docs/" kuratiert aufgearbeitet.

# Pl0.CLI

## Zweck

Die CLI stellt die Befehle `compile`, `run` und `run-pcode` bereit und verbindet Compiler und VM.

## Wichtige Bestandteile

- Optionen-Parser fuer Pascal-kompatible Compiler-Switches.
- Ausgabe von P-Code-Listen fuer didaktische Zwecke.
- Diagnoseausgaben fuer Fehler und Warnungen.

## Einstieg

```
dotnet run --project src/Pl0.Cli -- compile <datei.pl0> --out /tmp/example.pcode
```

# PL0.Core

## Zweck

PL0.Core enthaelt den Lexer, Parser und Compiler fuer PL/0 sowie die P-Code-Serialisierung.

## Verantwortlichkeiten

- Tokenisierung (Lexer) mit Positionsdaten.
- Syntaxanalyse (Parser) inkl. Symboltabelle.
- Codegenerierung (P-Code).
- Diagnoseobjekte fuer Fehler und Warnungen.

# PI0.Vm

## Zweck

PI0.Vm stellt die virtuelle Maschine fuer P-Code sowie I/O-Adapter bereit.

## Verantwortlichkeiten

- Interpreter fuer die P-Code-Instruktionen.
- Abstraktionen fuer Ein-/Ausgabe (Console/Buffered).
- Laufzeit-Diagnosen und Ergebnisobjekte.