

## **14 Zusammenfassung**

### **Teil 2 Fortgeschrittene Techniken**

---

## **15 Ownership im Detail**

### 15.1 Näheres zum bekannten Ownership-Modell

15.1.1 Move, Copy, Clone, Borrow

15.1.2 Lifetimes

15.1.3 Die Sicherheit von Rust

### 15.2 Smart Pointer

15.2.1 Box

15.2.2 Rc

15.2.3 RefCell und Cell

15.2.4 Zusammenfassung

### 15.3 Vergleich mit anderen Sprachen ohne Ownership

## **16 Nebenläufige und parallele Programmierung**

### 16.1 Grundlagen

### 16.2 Channels

### 16.3 Shared State

16.3.1 Arc

16.3.2 Send und Sync

16.3.3 Mutex

16.3.4 RwLock

### 16.4 Einfache Parallelisierung mit Rayon

### 16.5 Sicherheit trotz Parallelität

### 16.6 Async/Await

### 16.7 Zusammenfassung

## **17 Testen**

### 17.1 Arten von Tests

17.1.1 Unit-Tests

17.1.2 Integration-Tests

17.1.3 UI-Tests

- 17.1.4 Testpyramide, Nachwort
- 17.2 Rust, Cargo und Tests
  - 17.2.1 Platzierung von Testcode
- 17.3 Ausführung
  - 17.3.1 Erwartungen der Testergebnisse (Assertions)
  - 17.3.2 Benennung der Testfunktionen
- 17.4 Mocking
  - 17.4.1 Erste Schritte ohne Framework
  - 17.4.2 Einsatz eines Frameworks: Mockall
  - 17.4.3 Abschließendes zu Mockall
- 17.5 Snapshot-Tests mit `insta`
- 17.6 Der Rust-Compiler sieht viel, aber nicht alles
  - 17.6.1 Überläufe (Overflows)
  - 17.6.2 `OutOfBoundsCheck`
  - 17.6.3 Stockungen (Deadlocks)
- 17.7 Fazit

## **18 Webprogrammierung**

- 18.1 Einführung
  - 18.1.1 Warum Rust für Webprogrammierung?
  - 18.1.2 Warum nicht Rust für Webprogrammierung?
  - 18.1.3 Themen in diesem Kapitel
  - 18.1.4 Eine kleine Warnung vorab
- 18.2 Grundlagen von Rocket
  - 18.2.1 Handler
  - 18.2.2 Return Types
  - 18.2.3 Ein Blick hinter die Kulissen
  - 18.2.4 Shared State
- 18.3 Das Kontaktformular
  - 18.3.1 Routen
  - 18.3.2 Formulare
  - 18.3.3 Datenbankverbindung
  - 18.3.4 Was macht Rust bis hierher so besonders?

- 18.3.5 Middleware
- 18.3.6 Guards
- 18.3.7 Fairings oder Guards?
- 18.3.8 Serverside-Templates
- 18.3.9 Testen mit Rocket

## 18.4 Betrieb

- 18.4.1 Logging
- 18.4.2 Konfiguration
- 18.4.3 Deployment

## 18.5 Fazit

# **19 Microservices**

- 19.1 Eignet sich Rust für Microservices?
- 19.2 Aufteilung der Webanwendung in Microservices
- 19.3 Vorbereitungen
  - 19.3.1 Build mit Docker
  - 19.3.2 Cross Compilation
- 19.4 Die Microservices
  - 19.4.1 Anfragen annehmen: der »Web«-Service
  - 19.4.2 Gemeinsame Funktionalität
  - 19.4.3 Speichern der Anfragen in der Datenbank
  - 19.4.4 Mail verschicken
- 19.5 Betrieb
  - 19.5.1 Metriken und Monitoring
  - 19.5.2 Tracing
  - 19.5.3 Skalierung
- 19.6 Zusammenfassung

# **20 Systemnahe Programmierung**

- 20.1 Unsafe Rust
  - 20.1.1 Pointer-Grundlagen
  - 20.1.2 Unsafe in std: RefCell als Beispiel
- 20.2 Systemaufruf

- 20.2.1 Systemaufruf in Handarbeit
- 20.2.2 Systemaufruf mit dem Crate libc
- 20.3 Integration von externen Bibliotheken in Rust
  - 20.3.1 Fallstricke
- 20.4 Performanceuntersuchung
  - 20.4.1 Erste Schritte
  - 20.4.2 Benchmarks
  - 20.4.3 Untersuchungen
  - 20.4.4 Optimierung
- 20.5 Zusammenfassung

## **21 Spracherweiterungen (Language Bindings)**

- 21.1 Java
  - 21.1.1 Grundsätzliches – Java ruft Rust auf
  - 21.1.2 j4rs
  - 21.1.3 Zusammenfassung
- 21.2 Node.js
- 21.3 Fazit

## **22 WebAssembly**

- 22.1 Aktueller Stand von WebAssembly
  - 22.1.1 Im Browser
  - 22.1.2 Außerhalb des Browsers – ein Anfang
- 22.2 Rust & WASM
  - 22.2.1 Warum Rust für WASM?
  - 22.2.2 Im Browser: wasm-bindgen & wasm-pack
  - 22.2.3 Auf dem Server
- 22.3 Fazit

## **23 Zusammenfassung und Ausblick**

- 23.1 Zusammenfassung
- 23.2 Ausblick

## **Index**

# Vorwort

## Zielgruppe für dieses Buch

Dieses Buch richtet sich an Entwickler, die über den Tellerrand normaler Programmierung hinausschauen wollen. Mit Rust bietet sich die Möglichkeit, eine neue und interessante Programmiersprache kennenzulernen, die einiges anders macht als die gängigen Programmiersprachen.

## Notwendige Vorkenntnisse für das Buch

Sie sollten als Entwickler bereits Erfahrung in der objektorientierten Programmierung haben. Wissen um typische Konzepte gerade objektorientierter Sprachen sollte vorhanden sein. Wir erklären auch keine gängigen Programmierkonstrukte wie `if` oder `while`, sondern setzen die dafür notwendigen Kenntnisse voraus.

## Was dieses Buch nicht enthält

Dieses Buch wiederholt weder die leicht zu findenden Tutorials noch die exzellente Online-Dokumentation zu Rust. Es ist auch keine Einführung für Menschen, die das Programmieren erst lernen wollen (das Buch wäre sonst mehr als doppelt so dick).

## Was dieses Buch enthält

Unser Ziel ist es, Sie mit den Inhalten dieses Buchs in die Lage zu versetzen, auch komplexere praktische Programme mit Rust zu schreiben.

Natürlich hoffen wir außerdem, dass Sie genau wie wir die Freude dabei verspüren, Programme zu schreiben, bei denen Speicherlecks der Vergangenheit angehören.

Zum Schluss ist die Auseinandersetzung mit dem Ownership-Modell von Rust eine interessante Erfahrung, die uns auch in anderen Sprachen zu besseren Programmierern macht.