Backend Development with Rust – Complete Learning Guide

# 1. Rust Fundamentals (Must Know)

* • Ownership, borrowing, and lifetimes
* • Structs, enums, and pattern matching
* • Traits and generics
* • Error handling (Result, Option, ? operator)
* • Modules and packages (mod, use, crate)
* • Smart pointers (Box, Rc, Arc, Mutex)
* • Asynchronous programming (async/await, tokio, futures)
* • Iterators and closures

# 2. Asynchronous Programming in Rust

* • Learn the async/await syntax
* • Use the Tokio runtime (most popular async runtime)
* • Understand Futures and Concurrency (with tokio::spawn, join!)

# 3. Web Frameworks

* • Start with Axum (recommended)
* • Routing (GET, POST, PUT, DELETE)
* • Request and response handling
* • Middlewares
* • Extractors (path/query/body/form)
* • State management
* • Error handling in API routes
* • JSON API (serde integration)

# 4. Data Serialization

* • Use serde for serialization/deserialization
* • Use serde\_json for working with JSON

# 5. Databases and ORM

* • Use SQLx (async, compile-time checked SQL) or Diesel (sync, ORM)
* • Connecting to a DB (PostgreSQL or MySQL)
* • Querying data (SELECT, INSERT, UPDATE, DELETE)
* • Migrations
* • Pooling (sqlx::PgPool)
* • Environment config (dotenv, env!)

# 6. Authentication & Authorization

* • JWT (jsonwebtoken crate)
* • Password hashing with argon2 or bcrypt
* • Role-based access control

# 7. Testing

* • Unit tests with #[test]
* • Integration tests (test endpoints using HTTP clients)
* • Mocking

# 8. Building and Running the Server

* • Use cargo build, cargo run
* • .env management with dotenvy or config crate
* • Logging with tracing, log, or env\_logger
* • Dependency management in Cargo.toml

# 9. Deployment

* • Compile to a binary (cargo build --release)
* • Use systemd, Docker, or a shell script
* • Cross-compilation
* • Monitoring and metrics (Prometheus + OpenTelemetry)

# 10. Other Useful Concepts

* • File uploads
* • Websockets (tokio-tungstenite)
* • Background tasks (tokio::spawn)
* • Rate limiting
* • Email sending (lettre)
* • CI/CD (GitHub Actions, GitLab CI)

# Tools & Libraries to Explore

* • Web framework: axum, actix-web
* • Runtime: tokio
* • Database: sqlx, diesel, sea-orm
* • JSON: serde, serde\_json
* • Auth: jsonwebtoken, argon2
* • Config: dotenvy, config
* • Logging: tracing, log, env\_logger
* • Testing: tokio::test, reqwest
* • Email: lettre

# Suggested Learning Path

* • Week 1–2: Master Rust Basics
* • Week 3: Async Rust + Tokio
* • Week 4: Build with Axum
* • Week 5: Add DB support
* • Week 6: Add Auth (JWT + bcrypt)
* • Week 7: Add Features (file upload, email, pagination)
* • Week 8+: Testing, Docker, Deploy

# Bonus Projects (Practice)

* • Blog API: CRUD for posts with user login
* • Chat API: WebSocket server
* • Inventory App: Products, orders, sales with reports
* • Quiz App: Questions, answers, scoring logic