Jean Jacques Avril

Funktionale Programmierung

Mit Go und Dart im Vergleich

Am Beispiel eines Zeiterfassungs-Tools

Agenda

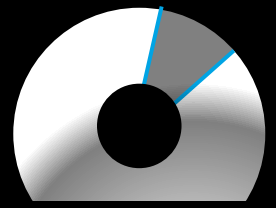
1. Einführung: Motivation & Idee
2. Demo & Umsetzung
3. FP-Sprachfeatures und Eignung
4. FP-Bibliotheken und Erweiterungen
5. Fazit und Ausblick

Motivation

- Stabilerer Code
- Besseres Fehlerhandling
- Persönliche Herausforderung

und Idee

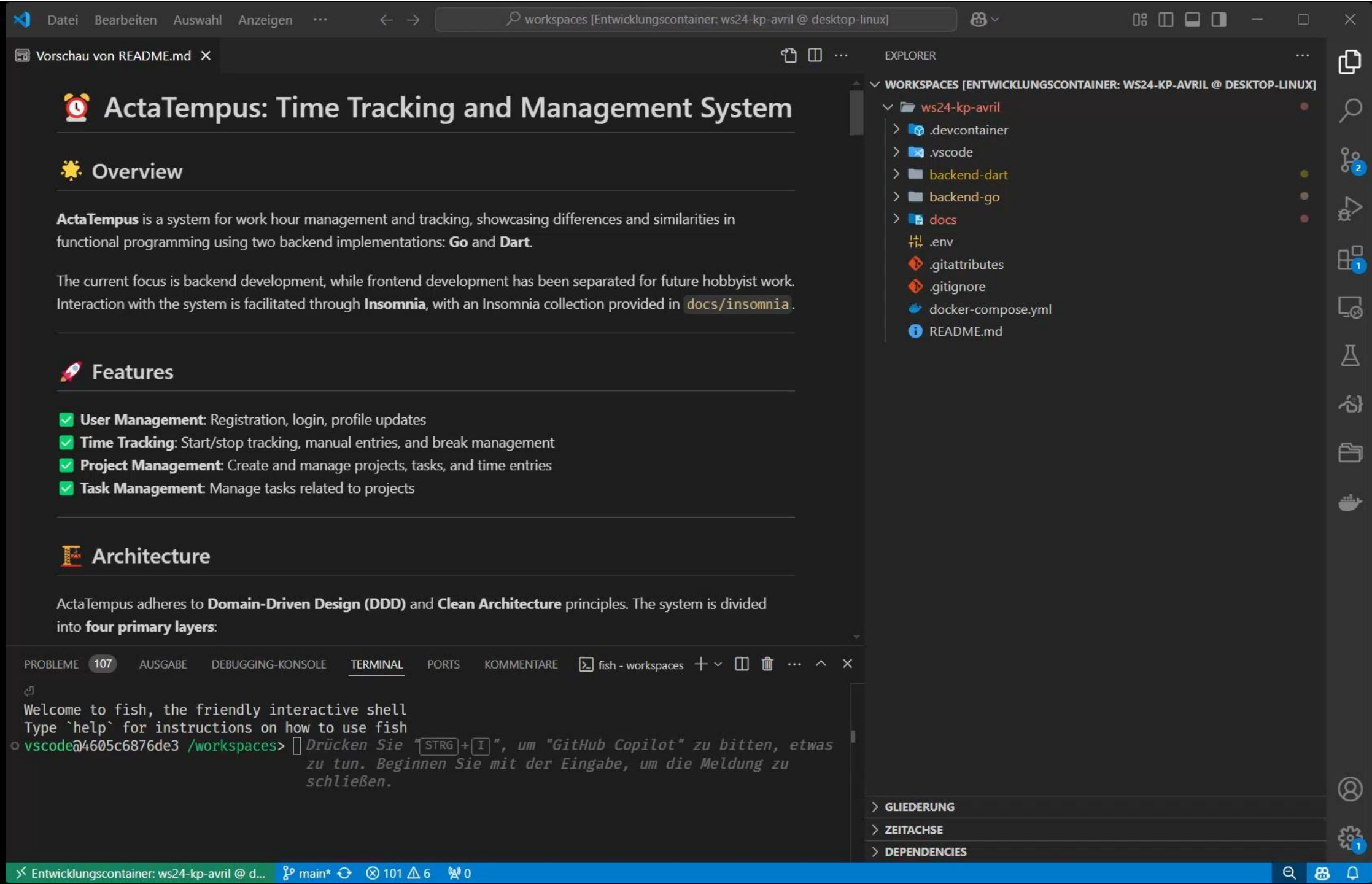
Es fehlt ein modernes Open-Source-Tool für Arbeitszeiterfassung.



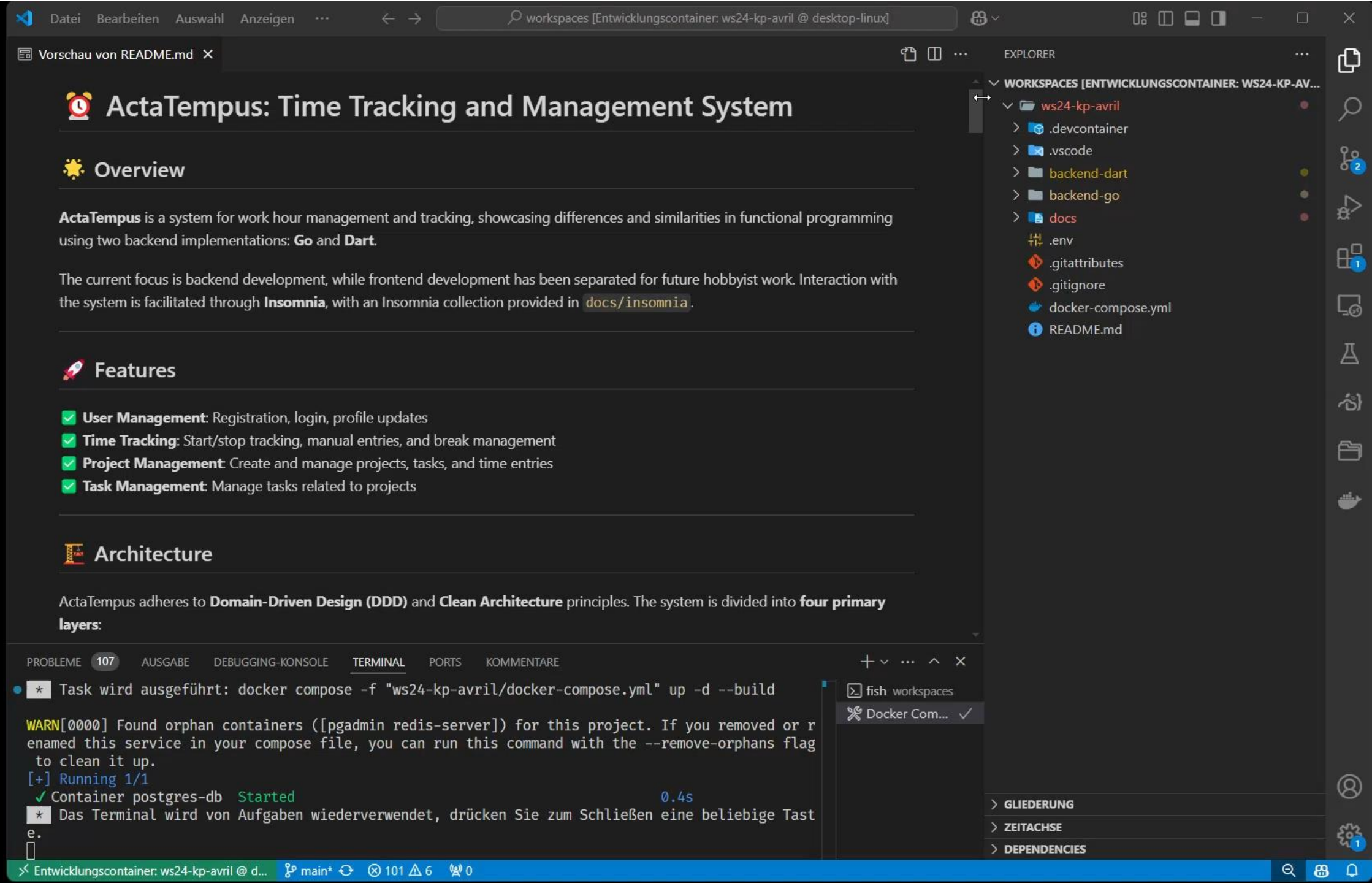
ActaTempus

ActaTempus löst dieses Problem mit Backends in Go und Dart, basierend auf funktionaler Programmierung.

1. Datenbank Setup: Postges starten



1. Datenbank Setup: Codegen & Schema



2. Start Go Backend

Datei Bearbeiten Auswahl Anzeigen ··· workspaces [Entwicklungscontainer: ws24-kp-avril @ desktop-linux]

Vorschau von README.md

```
docker compose up
```

Deploy the Schema

Apply the schema using Prisma. Only one backend needs to execute the command, as both backends share the schema:

```
cd backend-go
go run github.com/steebchen/prisma-client-go db push
```

or

```
cd backend-dart
bunx prisma db push
```

Prisma Studio

Launch Prisma Studio for database visualization and management:

```
cd backend-dart
bunx prisma studio
```

EXPLORER

WORKSPACES [ENTWICKLUNGSCONTAI...

- ws24-kp-avril
  - .devcontainer
  - .vscode
  - backend-dart
  - backend-go
  - docs
  - .env
  - .gitattributes
  - .gitignore
  - docker-compose.yml
  - README.md

PROBLEME 107 AUSGABE DEBUGGING-KONSOLE TERMINAL PORTS KOMMENTARE

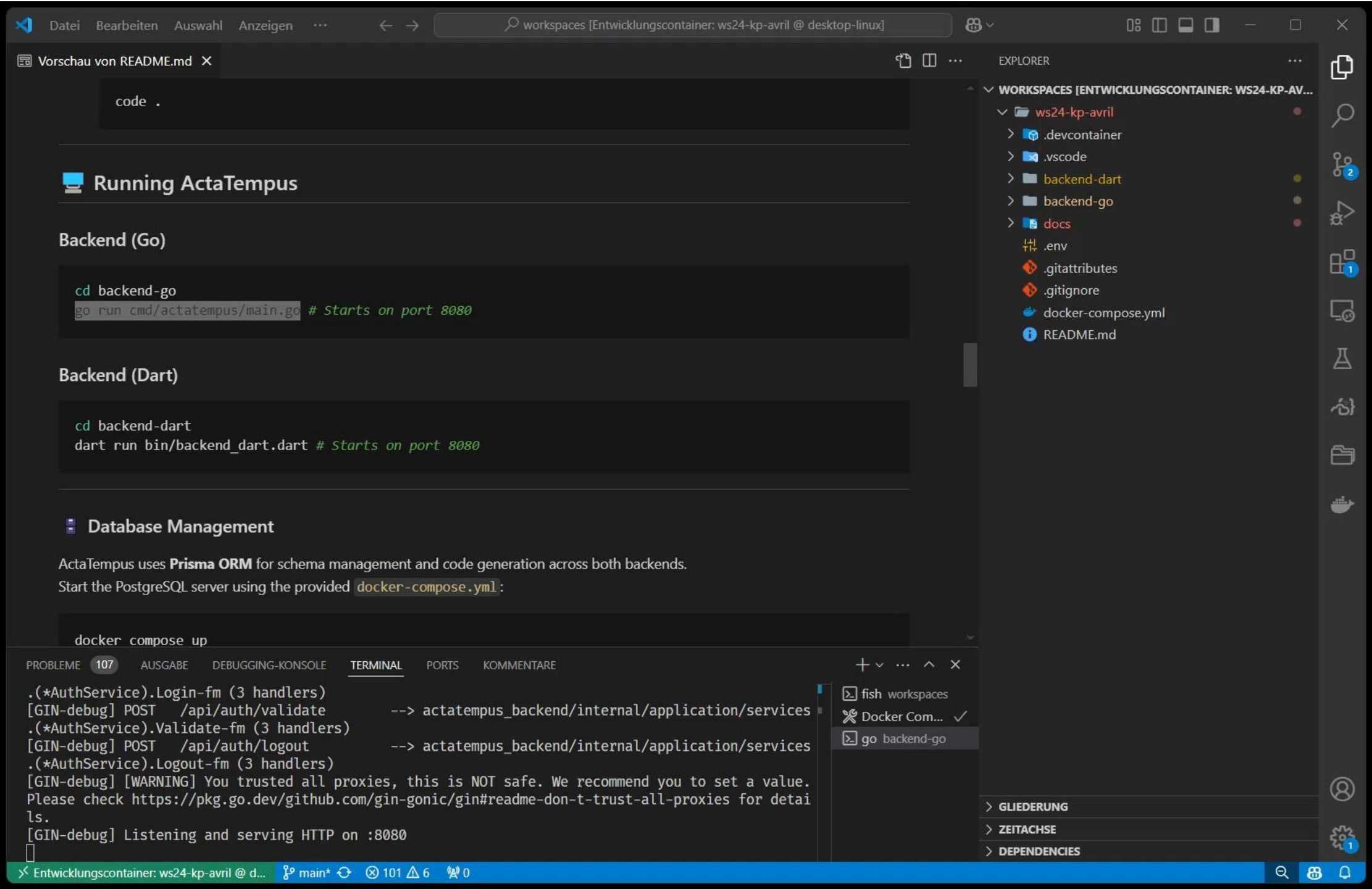
```
Environment variables loaded from .env
Prisma schema loaded from prisma/schema.prisma
Datasource "db": PostgreSQL database "time_tracking_db", schema "public" at "host.docker.internal
:5432"

The database is already in sync with the Prisma schema.

✓ Generated Prisma Client Go to ./internal/infrastructure/data/db in 2.08s

vscode@4605c6876de3 /w/w/backend-go (main)>
```

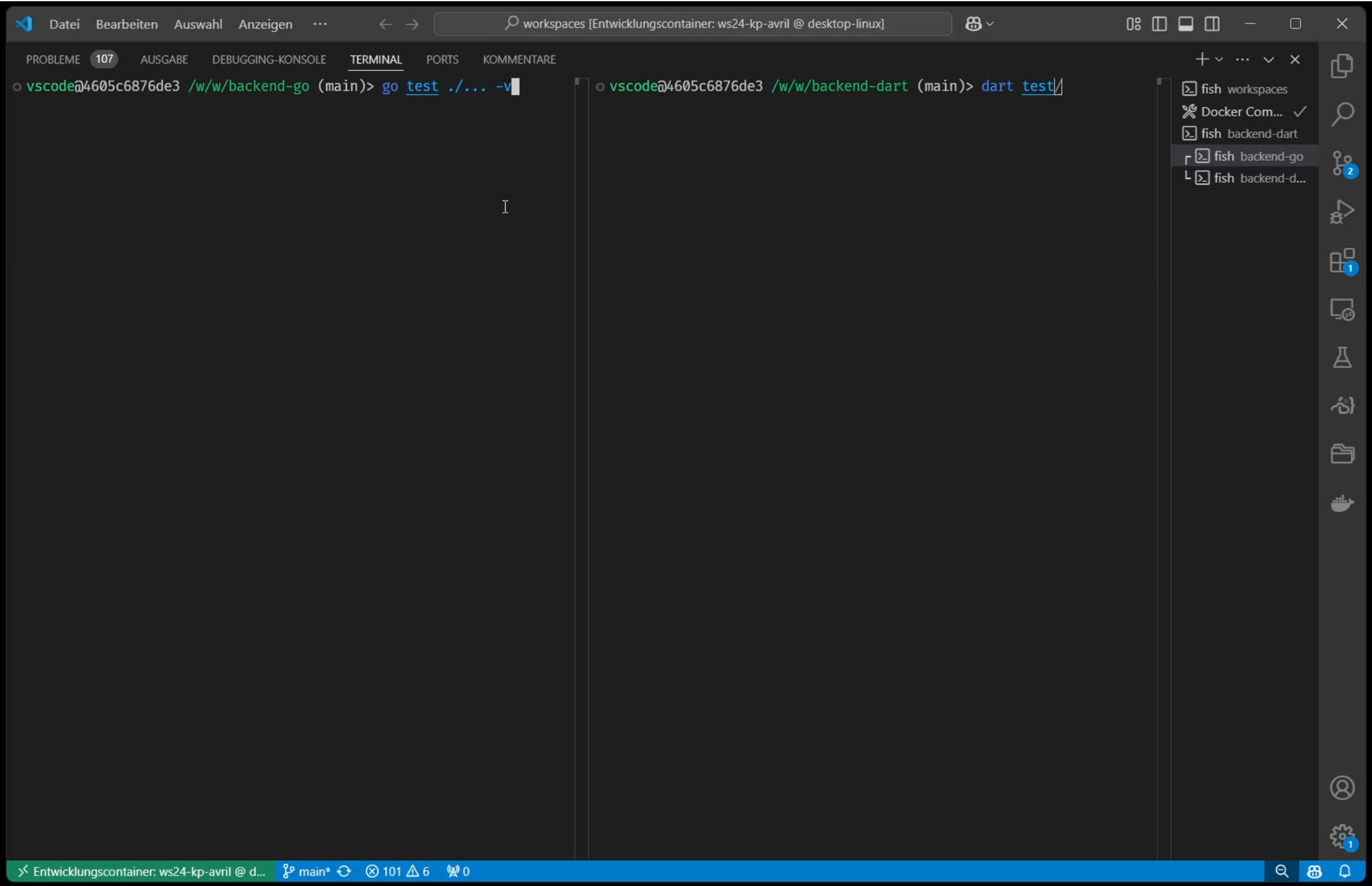
fish workspaces
Docker Com... ✓
fish backend-go

GLIEDERUNG
ZEITACHSE
DEPENDENCIES

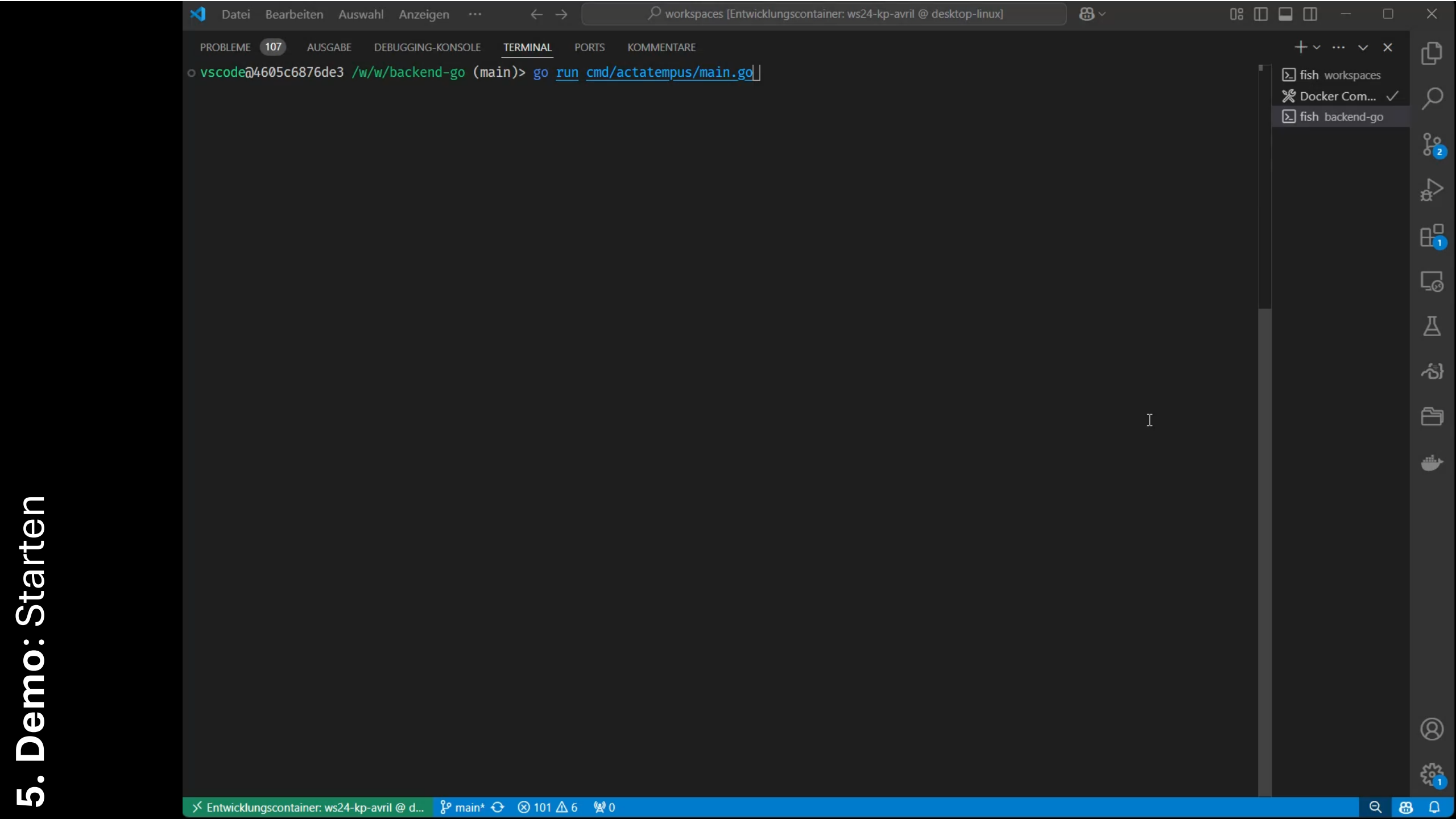
Entwicklungscontainer: ws24-kp-avril @ d... main* 101 6 0

3. Start Dart Backend



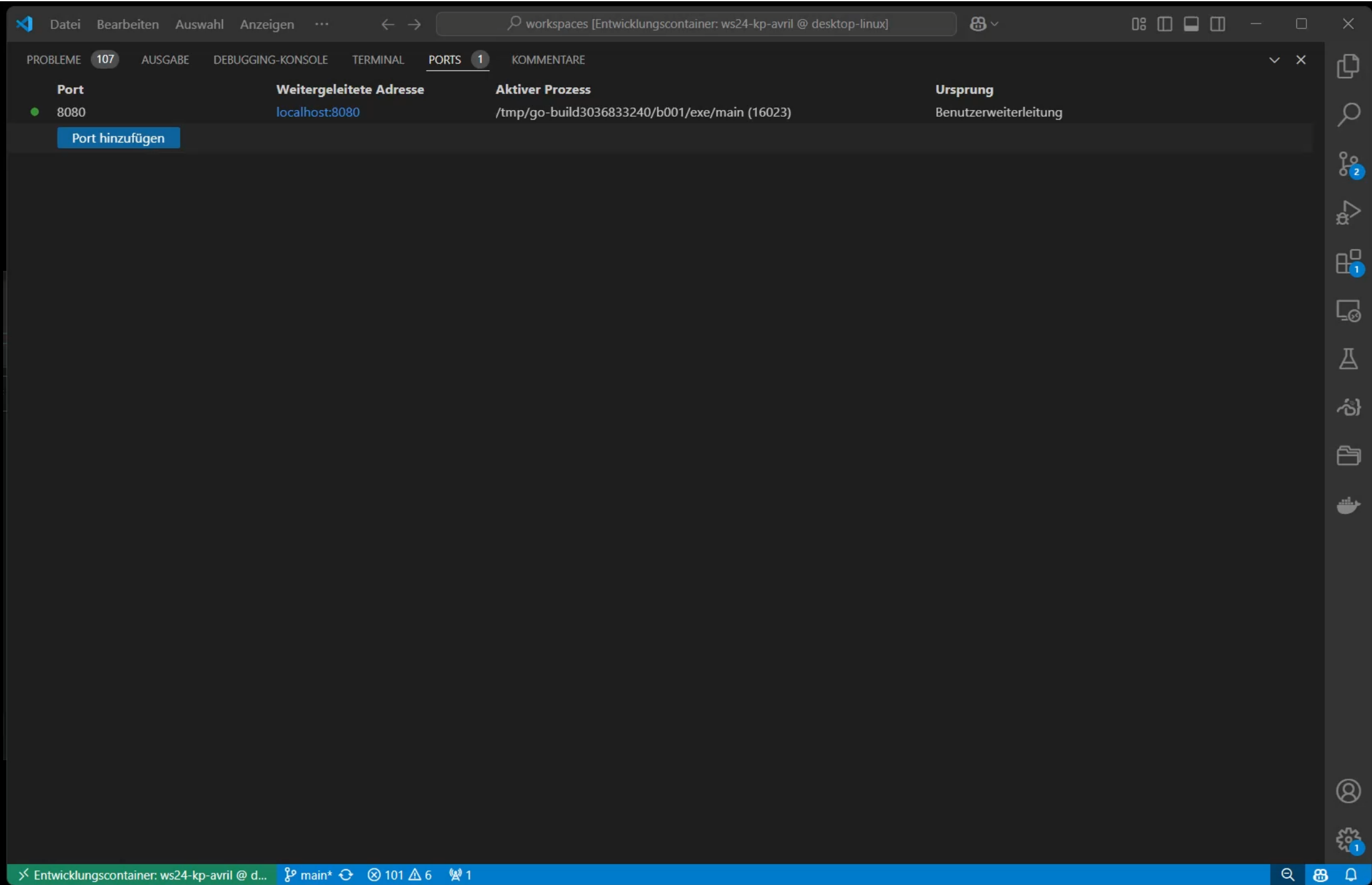

Datei Bearbeiten Auswahl Anzeigen ···
workspaces [Entwicklungscontainer: ws24-kp-avril @ desktop-linux]
PROBLEME 107 AUSGABE DEBUGGING-KONSOLE TERMINAL PORTS KOMMENTARE
vscode@4605c6876de3 /w/w/backend-go (main)> go test ./... -v
vscode@4605c6876de3 /w/w/backend-dart (main)> dart test
fish workspaces
Docker Com...
fish backend-dart
fish backend-go
fish backend-d...
Entwicklungscontainer: ws24-kp-avril @ d...
main*
101 6
0

4. Tests

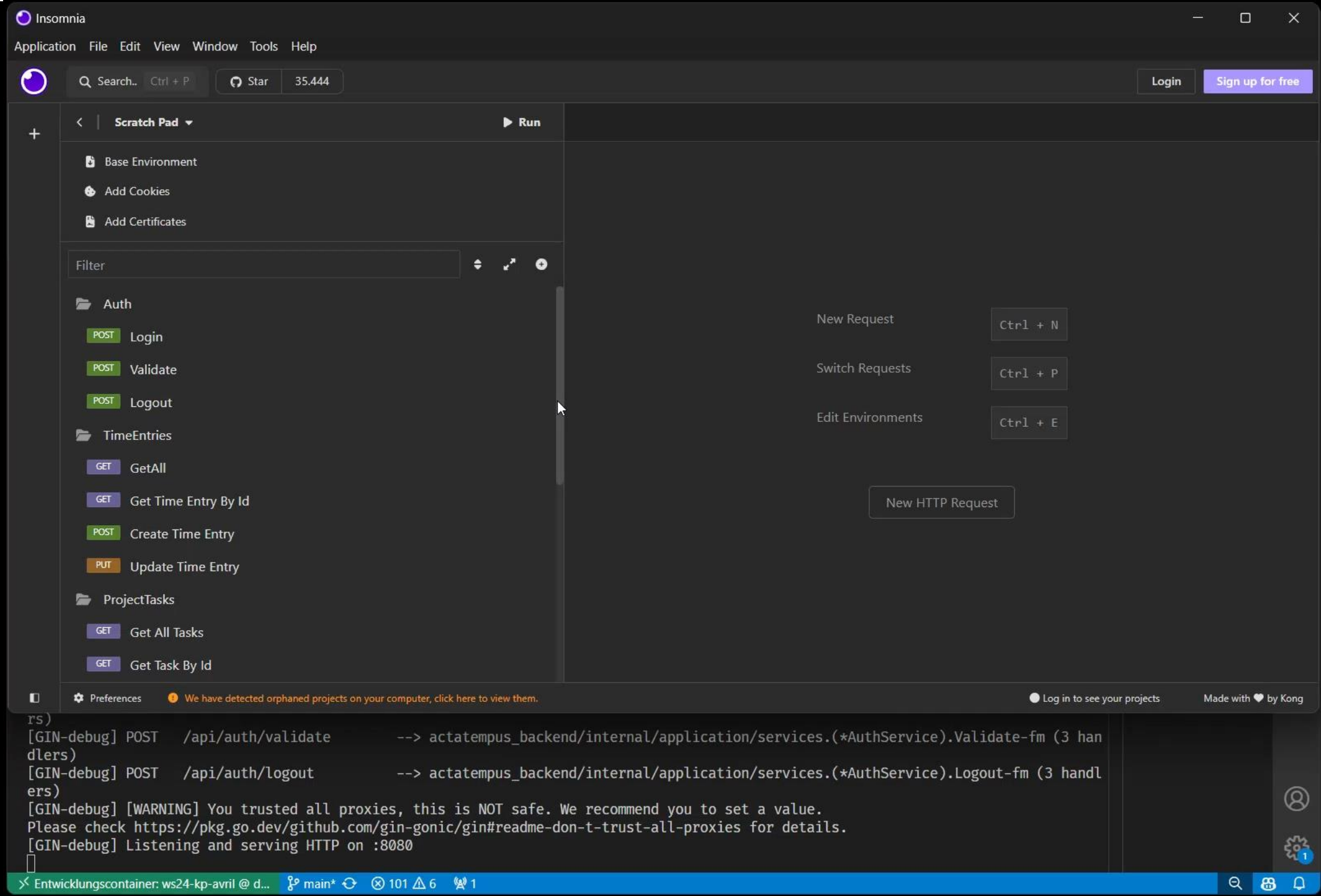


5. Demo: Starten

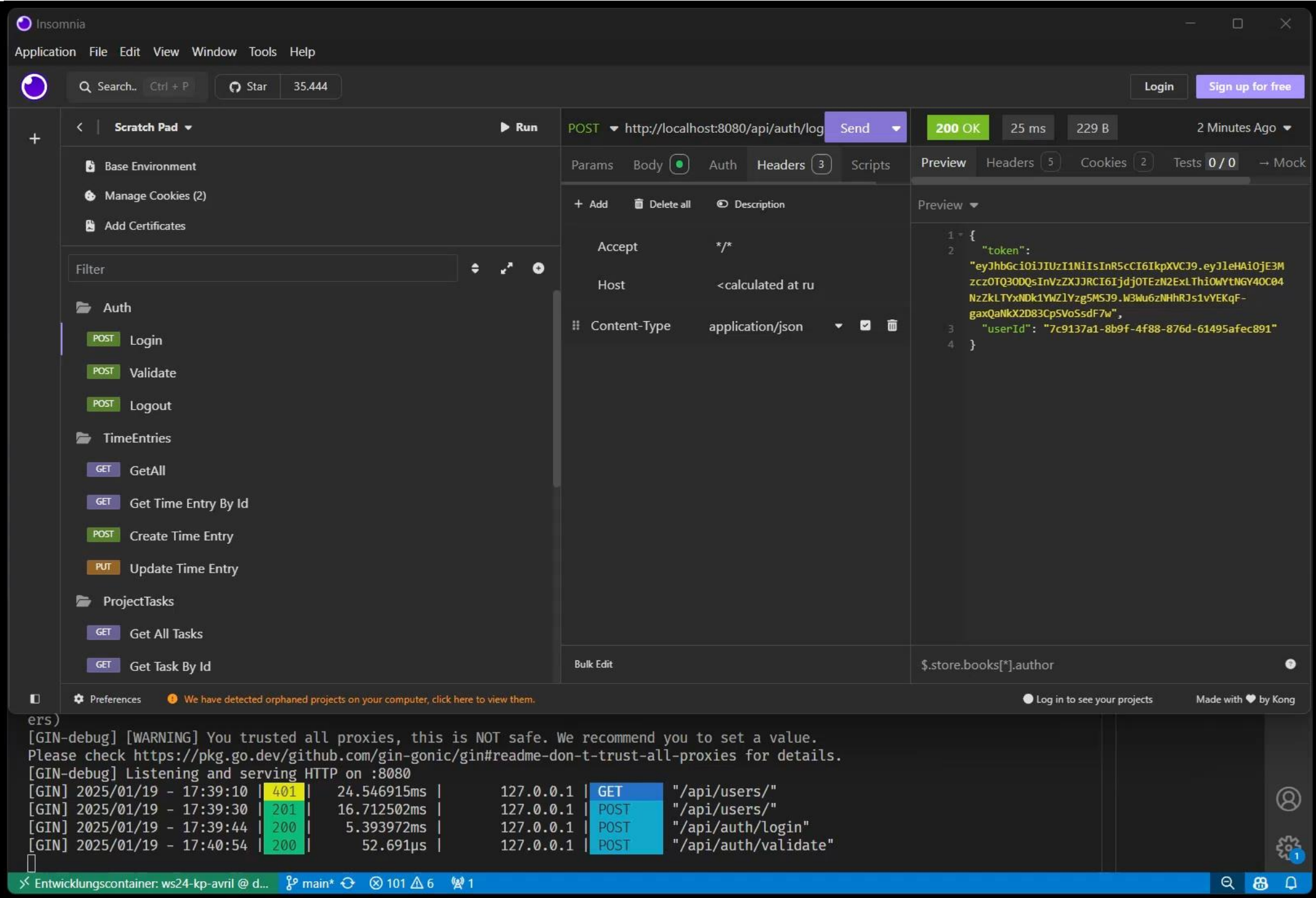
4. 5. Demo: API-Client Setup(Insomnia)



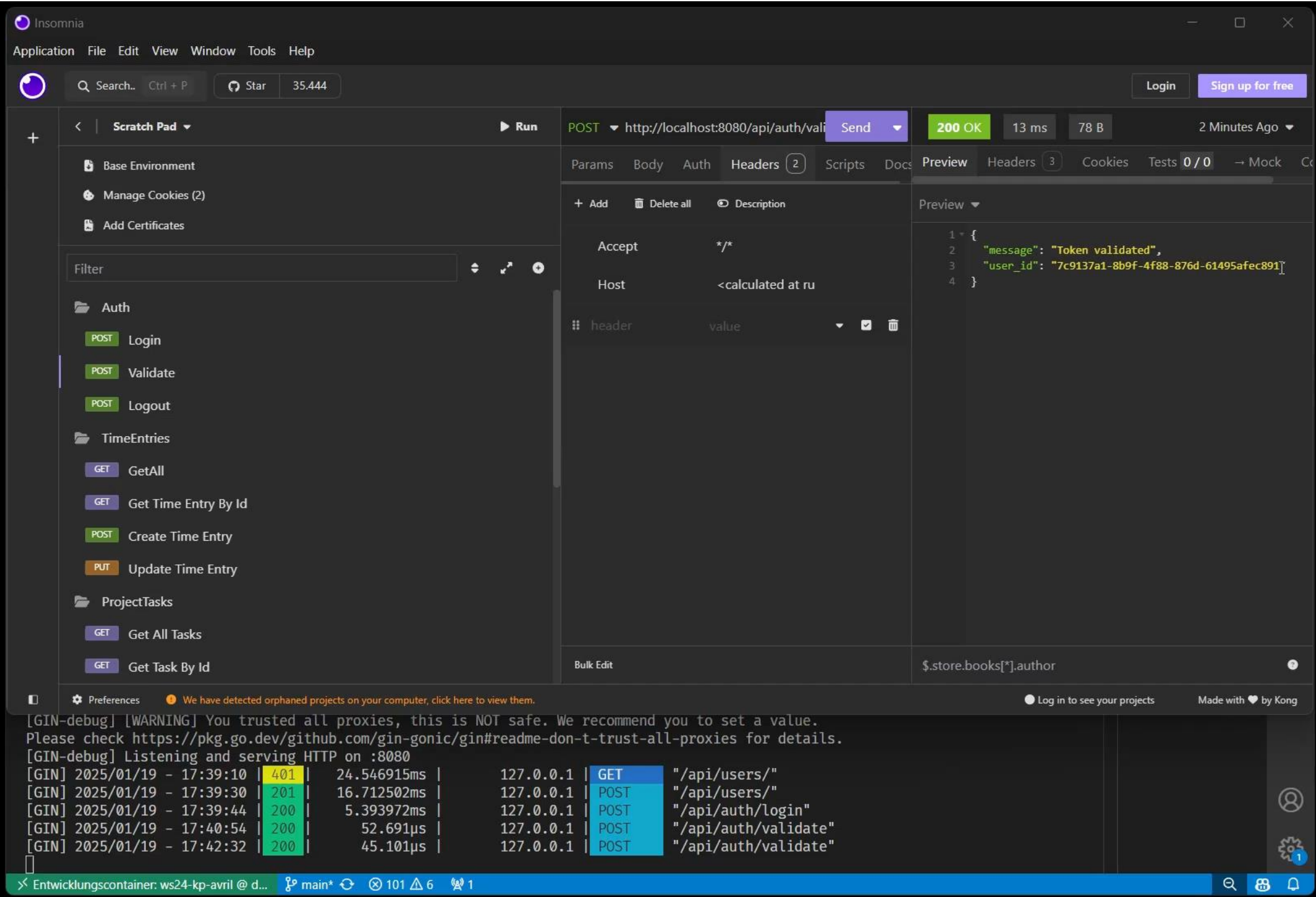
4. 5. Demo: Benutzer anlegen



4. 5. Demo: Token verifizieren



4. 5. Demo: Zeiteinträge anlegen

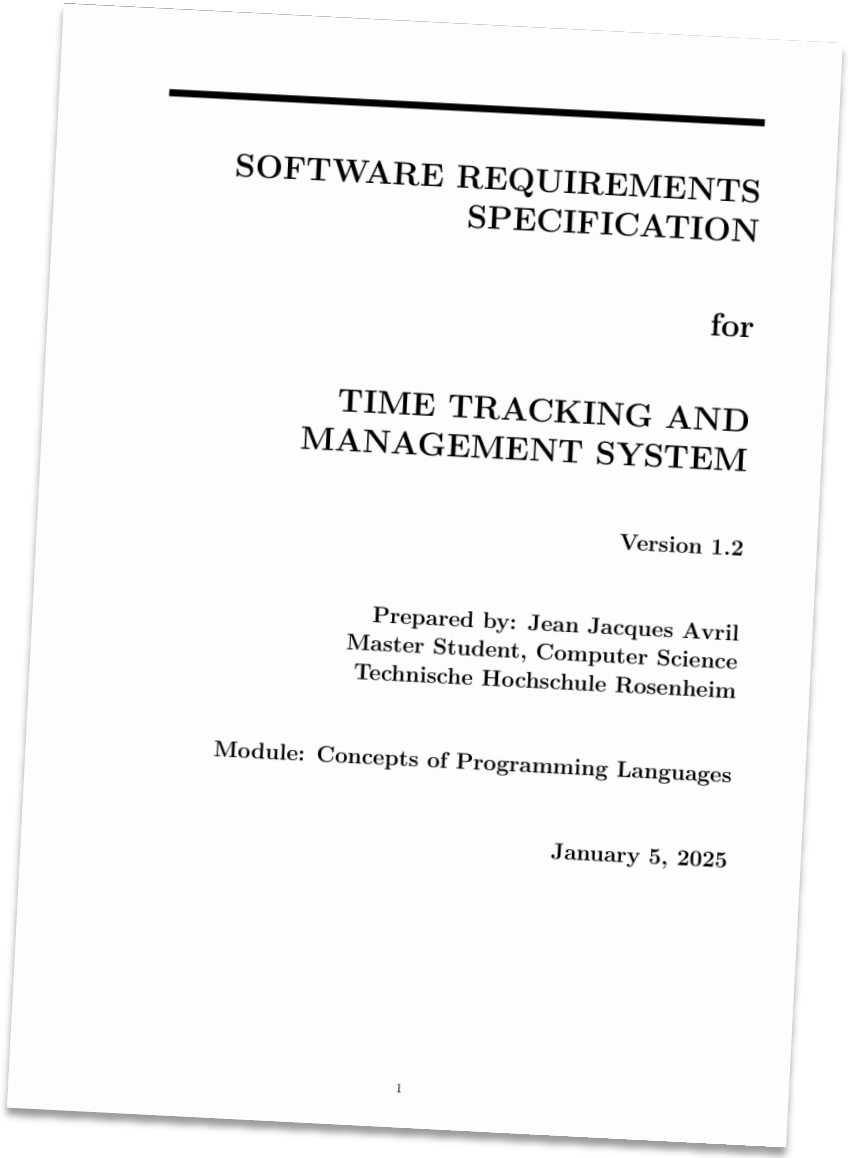


Umsetzung

Iterative Entwicklung

Spezifikation

1. Einführung: Ziel & Scope
2. Anforderungen
3. Technologie und Domänenwissen
4. System Design: Architektur, Module, Datenbank
5. Implementierungsdetails Go Backend
6. Implementierungsdetails Dart Backend
7. Definition der API-Endpunkte



SOFTWARE REQUIREMENTS SPECIFICATION

for

TIME TRACKING AND MANAGEMENT SYSTEM

Version 1.2

Prepared by: Jean Jacques Avril
Master Student, Computer Science
Technische Hochschule Rosenheim

Module: Concepts of Programming Languages

January 5, 2025

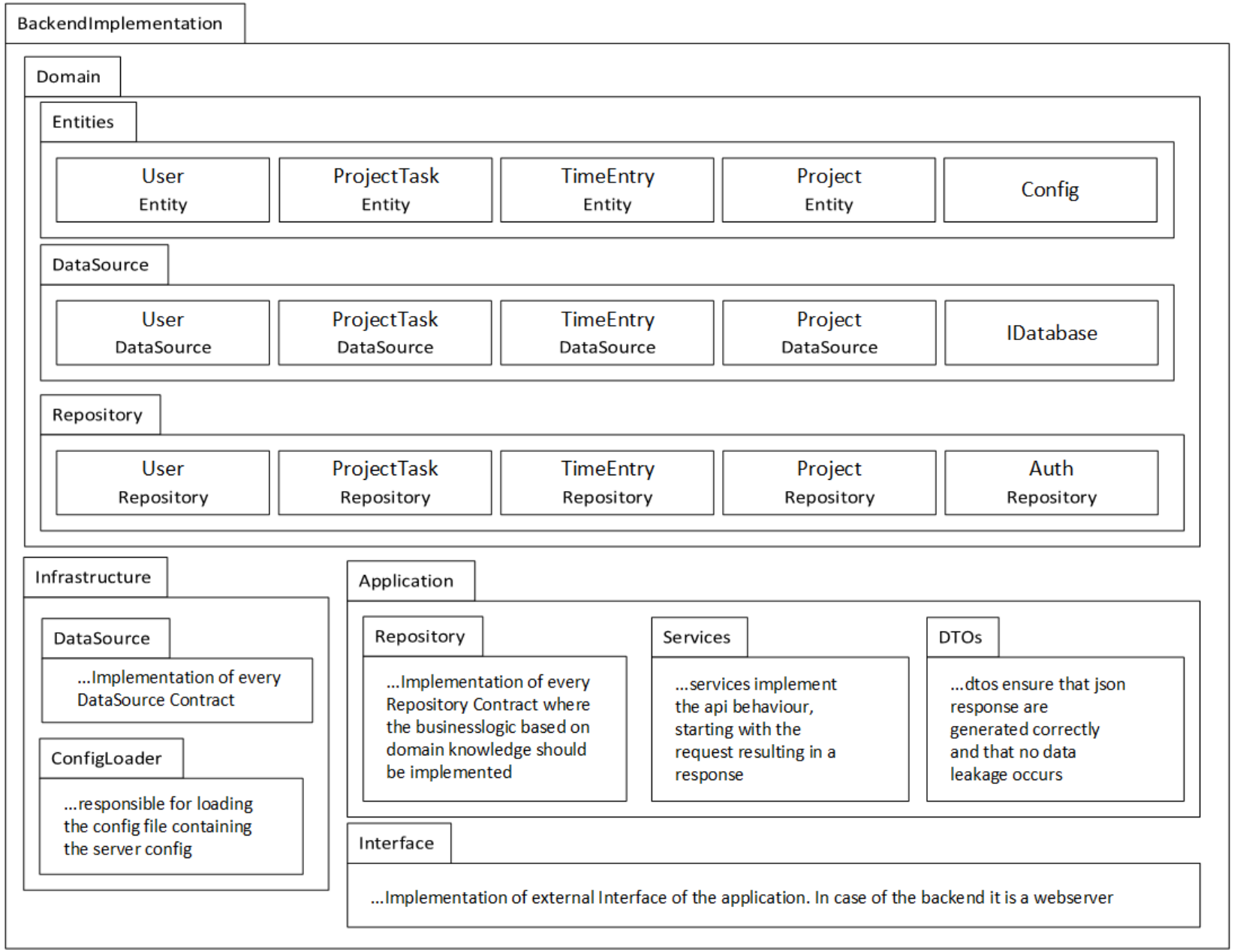
1

https://inf-git.fh-rosenheim.de/studavrije7683/ws24-kp-avril/-/raw/main/docs/KP_Spec-4.pdf

Umsetzung

DDD Architektur

Domain
Driven
Design



FP Sprachfeatures und Eignung

Benötigtes FP-Feature	Dart	Go
Erstklassige Funktionen	✅ Unterstützt	✅ Unterstützt
High-Order Functions	✅ Unterstützt	✅ Unterstützt
Closures	✅ Unterstützt	✅ Unterstützt
Generics	✅ Unterstützt (Null Safety integriert)	✅ Unterstützt (ab Go 1.18)
Statische Typisierung	✅ Unterstützt	✅ Unterstützt
Nebenläufigkeit	✅ Nativ (Future, Stream)	✅ Nativ (Goroutines, Channels)
Rekursion*	✅ Unterstützt	✅ Unterstützt
Pattern Matching	✅ Unterstützt (ab Dart 3.0)	❌ Nicht unterstützt

***ohne Tail Call Optimization**: Bei vielen Rekursionen Stack-Overflow möglich

FP Sprachfeatures und Eignung

Benötigtes FP-Feature	Dart	Go
Pattern Matching	✅ Unterstützt (ab Dart 3.0)	❌ Nicht unterstützt
Immutability	❌ Nicht nativ, nur über Konvention	❌ Nicht nativ, nur über Konvention
Lazy Evaluation	❌ Nicht nativ, manuell möglich	❌ Nicht nativ, manuell möglich
Currying	❌ Nicht nativ, manuell möglich	❌ Nicht nativ, manuell möglich
Monaden	❌ Nicht nativ, manuell möglich	❌ Nicht nativ, manuell möglich
Funktoren	❌ Nicht nativ, manuell möglich	❌ Nicht nativ, manuell möglich
Funktionale Komposition	❌ Nicht nativ, manuell möglich	❌ Nicht nativ, manuell möglich

FP Sprachfeatures und Eignung

Benötigtes FP-Feature | Dart | Co

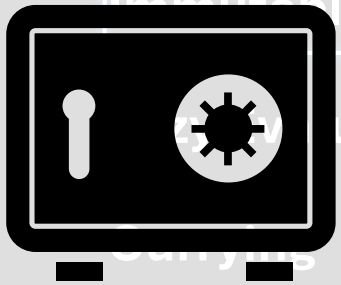
Pattern Mat

Immutabilit

Monaden

Funktoren

Funktionale Komposition



Dart Immutability-Beispiel

```dart
final List<int> numbers = [1, 2, 3];
numbers.add(4); // Das ist erlaubt
// numbers = [5, 6, 7]; // Fehler: Die Referenz kann nicht geändert werden
print(numbers); // [1, 2, 3, 4]

const List<int> numbers = [1, 2, 3];
// numbers.add(4); // Fehler: `numbers` ist unveränderlich
print(numbers); // [1, 2, 3]
```

❌ Nicht nativ, manuell möglich | ❌ Nicht nativ, manuell möglich

Für echte Immutability wird ein Copy-On-Write Feature benötigt!

*ohne Tail Call Optimization: Bei vielen Rekursionen Stack-Overflow möglich

FP Sprachfeatures und Eignung

Vorläufiges Fazit

Dart und **Go** bieten solide **Grundlagen** für funktionale Programmierung. Bei der praktischen Anwendung gibt es einige **Einschränkungen,** da oft eine manuelle Implementierung erforderlich ist.

Lösung: Zusätzliche FP-Bibliotheken und Erweiterungen

FP-Bibliotheken und Erweiterungen

Dart

fpdart

https://github.com/SandroMaglione/fpdart

- Monaden (Option, Either, Task)
- Funktoren
- Funktionale Komposition
- Lazy Evaluation (Task, TaskEither)
- Immutability

freezed

https://github.com/rrousselGit/freezed

- Immutability

Go

FP-GO

https://github.com/IBM/fp-go

- Monaden (Option, Either, Task)
- Funktoren
- Funktionale Komposition
- Currying

Gelöst Teilweise gelöst Nicht gelöst

FP Sprachfeatures und Eignung

Immutability mit Freezed

```dart
@freezed
class ProjectCreate with _$ProjectCreate {
  const factory ProjectCreate({
    String? id,
    required String name,
    String? description,
    String? clientId,
    required String userId,
  }) = _ProjectCreate;
}
```

```dart
@override
TaskEither<IError, ProjectTask> create(ProjectTaskCreate task) {
  return database.tasks
      .generateId()
      .map((id) => task.copyWith(id: id))
      .flatMap(database.tasks.create);
}
```

FP Sprachfeatures und Eignung

Benötigtes FP-Feature	Dart	Go
Pattern Matching	✅ Unterstützt (ab Dart 3.0)	❌ Nicht unterstützt
Immutability	✅ Unterstützt mit freezed	❌ Nicht nativ, nur über Konvention
Lazy Evaluation	✅ Unterstützt mit fpdart	❌ Nicht nativ, manuell möglich
Currying	❌ Nicht nativ, manuell möglich	✅ Unterstützt mit fp-go
Monaden	✅ Unterstützt mit fpdart	✅ Unterstützt mit fp-go
Funktoren	✅ Unterstützt mit fpdart	✅ Unterstützt mit fp-go
Funktionale Komposition	✅ Unterstützt mit fpdart	✅ Unterstützt mit fp-go

FP Sprachfeatures und Eignung

Benötigtes FP-Feature

- Pattern Matching
- Immutability
- Lazy Evaluation
- Currying
- Monaden
- Funktoren
- Funktionale Komposition

```
  @Route.get('/')
  Future<Response> listProjects(Request request) async {
    return checkAuth(request, authRepository)
        .flatMap((_) ⇒ projects.findAll())
        .flatMap(mapper.listTo)
        .map((projects) ⇒ projects.map((project) ⇒ project.toJson()).toList())
        .map(jsonEncode)
        .toResponse()
        .run();
  }

extension TaskEitherResponseExtensionsFromString on TaskEither<IError, String> {
  Task<Response> toResponse() ⇒ match(
        ResponseHelpers.fromError,
        Response.ok,
      );
}
```

FP Sprachfeatures und Eignung

Dart Currying-Beispiel

```dart
Function(A) Function(B) curry<A, B, R>(R Function(A, B) f) {
  return (A a) ⇒ (B b) ⇒ f(a, b);
}

void main() {
  int add(int x, int y) ⇒ x + y;

  final curriedAdd = curry(add);

  final add5 = curriedAdd(5);

  final result = add5(3);

  print(result); // 8
}
```

Go Currying-Beispiel

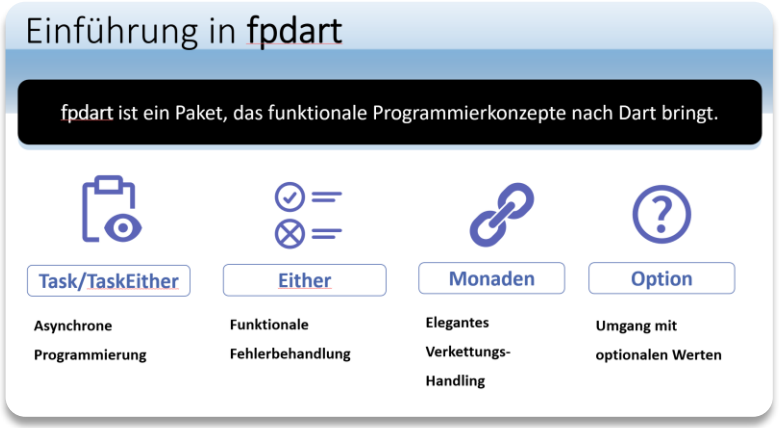
```go
package main
import "fmt"
func curry[A, B, R any](f func(A, B) R) func(A) func(B) R {
    return func(a A) func(B) R {
        return func(b B) R {
            return f(a, b)
        }
    }
}

func main() {
    add := func(x, y int) int {
        return x + y
    }
    curriedAdd := curry(add)
    add5 := curriedAdd(5)
    result := add5(3)
    fmt.Println(result) // 8
}
```

FP Sprachfeatures und Eignung

Benötigtes FP-Feature	Dart	Go
Pattern Matching	✅ Unterstützt (ab Dart 3.0)	❌ Nicht unterstützt
Immutability	✅ Unterstützt mit freezed	❌ Nicht nativ, nur über Konvention
Lazy Evaluation	✅ Unterstützt mit fpdart	❌ Nicht nativ, manuell möglich
Currying	❌ Nicht nativ, manuell möglich	✅ Unterstützt mit fp-go
Monaden	✅ Unterstützt mit fpdart	✅ Unterstützt mit fp-go
Funktoren	✅ Unterstützt mit fpdart	✅ Unterstützt mit fp-go
Funktionale Komposition	✅ Unterstützt mit fpdart	✅ Unterstützt mit fp-go

FP Sprachfeatures und Eignung

Erinnerung: Zwischenpräsentation

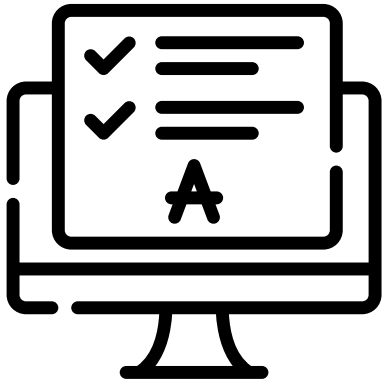



```dart
@Route.get('/')
Future<Response> listEntries(Request request) async {
  return checkAuth(request, authRepository) // TaskEither<IError, String>
      .flatMap(
          (_) => entries.findAll()) // TaskEither<IError, List<TimeEntry>>
      .flatMap(mapper.listTo) // TaskEither<IError, List<TimeEntryDto>>
      .map((entries) => entries
          .map((entry) => entry.toJson())
          .toList()) // TaskEither<IError, List<Map<String, dynamic>>>
      .map(jsonEncode) // TaskEither<IError, String>
      .toResponse() // Task<Response>
      .run(); // Future<Response>P
}

extension TaskEitherResponseExtensionsFromString on TaskEither<IError, String> {
  Task<Response> toResponse() => match(
        ResponseHelpers.fromError,
        Response.ok,
      );
}
```

FP Sprachfeatures und Eignung

```go
import (
    A "github.com/IBM/fp-go/array"
    E "github.com/IBM/fp-go/either"
    F "github.com/IBM/fp-go/function"
    "github.com/gin-gonic/gin"
)


func (s *TimeEntryService) GetAllTimeEntries(c *gin.Context) {
    F.Pipe3(
        CheckAuth(c, s.authRepository),
        E.Chain(F.Constant1[string](s.repository.FindAll(c.Request.Context()))),
        E.Map[error](A.Map(mappers.MapTimeEntryToDTO)),
        E.Fold(
            HandleError(c),
            HandleSuccess[[]dto.TimeEntryDTO](c, http.StatusOK),
        ),
    )
}
```

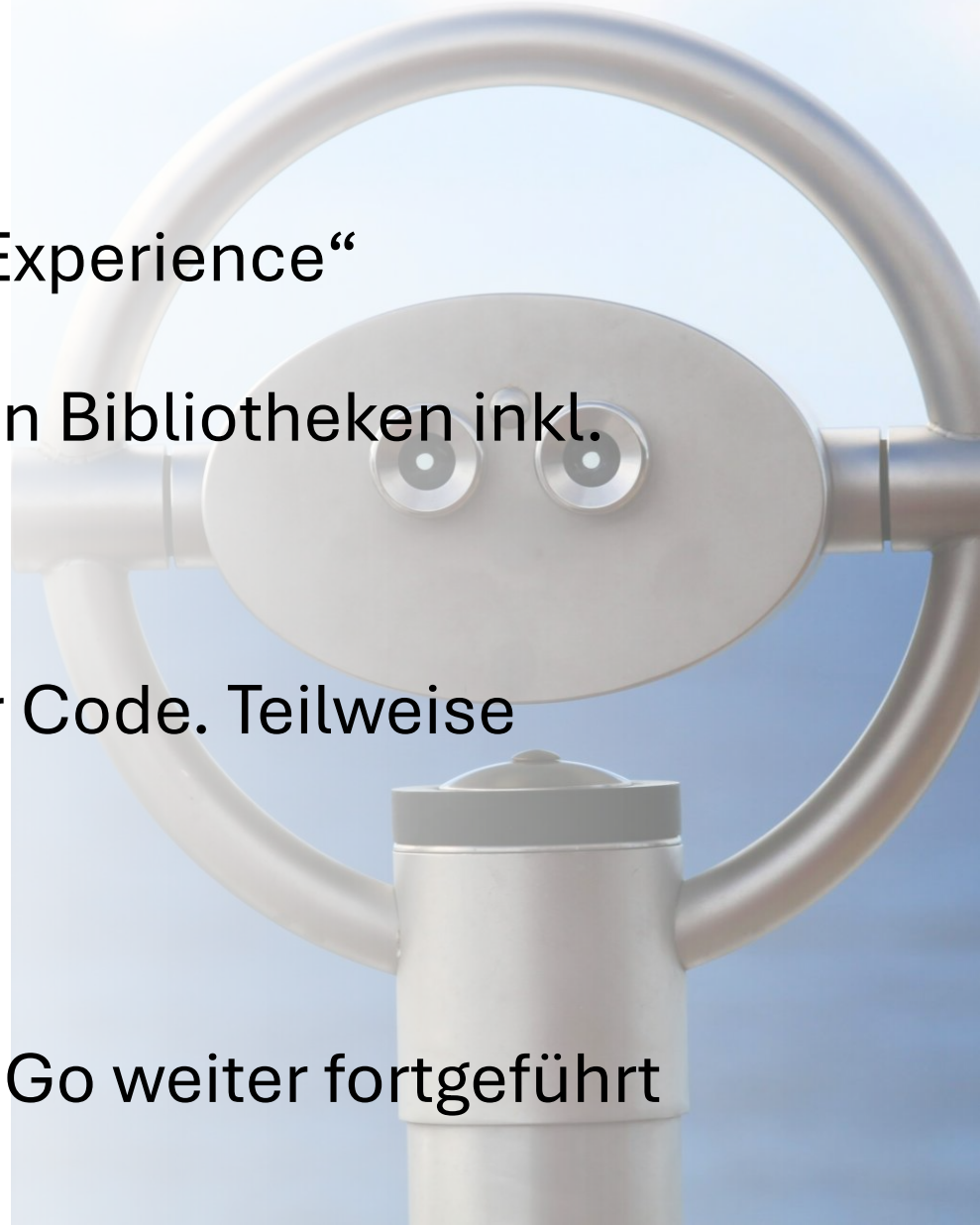
Fazit und Ausblick



Überaus positive „Entwickler-Experience“

Voraussetzung: Einarbeitung in Bibliotheken inkl. Eingewöhnungsphase

Ergebnis: Sehr übersichtlicher Code. Teilweise entstehen Mischformen.



Ausblick: Projekt wird als Hobbyprojekt in Go weiter fortgeführt