

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

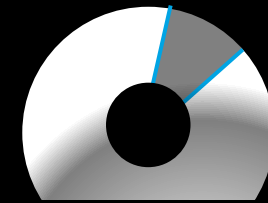
Fazit und Ausblick

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

The screenshot shows the Visual Studio Code interface with the ActaTempus project open. The main editor displays the README.md file, which provides an overview and features of the system. The Explorer view on the right shows the project structure, including folders for backend development and documentation. The terminal at the bottom shows the fish shell prompt.

Vorschau von README.md

ActaTempus: Time Tracking and Management System

Overview

ActaTempus is a system for work hour management and tracking, showcasing differences and similarities in functional programming using two backend implementations: **Go** and **Dart**.

The current focus is backend development, while frontend development has been separated for future hobbyist work. Interaction with the system is facilitated through **Insomnia**, with an Insomnia collection provided in [docs/insomnia](#).

Features

- ✓ **User Management:** Registration, login, profile updates
- ✓ **Time Tracking:** Start/stop tracking, manual entries, and break management
- ✓ **Project Management:** Create and manage projects, tasks, and time entries
- ✓ **Task Management:** Manage tasks related to projects

Architecture

ActaTempus adheres to **Domain-Driven Design (DDD)** and **Clean Architecture** principles. The system is divided into **four primary layers**:

WORKSPACES [ENTWICKLUNGSCONTAINER: WS24-KP-AVRIL @ DESKTOP-LINUX]

- 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 | fish - workspaces

```
Welcome to fish, the friendly interactive shell
Type `help` for instructions on how to use fish
vscodex@4605c6876de3 /workspaces> 
```

Drücken Sie "STRG" + "I", um "GitHub Copilot" zu bitten, etwas zu tun. Beginnen Sie mit der Eingabe, um die Meldung zu schließen.

GLIEDERUNG
ZEITACHSE
DEPENDENCIES

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

1. Datenbank Setup: Codegen & Schema

The screenshot displays the Visual Studio Code interface with the ActaTempus project open. The main editor shows the README.md file, which provides an overview, features, and architecture of the system. The Explorer sidebar on the right lists the project's workspace structure, including folders for .devcontainer, .vscode, backend-dart, backend-go, docs, and files like .env, .gitattributes, .gitignore, docker-compose.yml, and README.md. The bottom panel shows the Terminal with the output of a Docker Compose command, indicating that the postgres-db container has started successfully. The status bar at the bottom indicates the current workspace is 'ws24-kp-avril' and the active branch is 'main'.

Vorschau von README.md

ActaTempus: Time Tracking and Management System

Overview

ActaTempus is a system for work hour management and tracking, showcasing differences and similarities in functional programming using two backend implementations: **Go** and **Dart**.

The current focus is backend development, while frontend development has been separated for future hobbyist work. Interaction with the system is facilitated through **Insomnia**, with an Insomnia collection provided in `docs/insomnia`.

Features

- ✓ **User Management:** Registration, login, profile updates
- ✓ **Time Tracking:** Start/stop tracking, manual entries, and break management
- ✓ **Project Management:** Create and manage projects, tasks, and time entries
- ✓ **Task Management:** Manage tasks related to projects

Architecture

ActaTempus adheres to **Domain-Driven Design (DDD)** and **Clean Architecture** principles. The system is divided into **four primary layers**:

PROBLEME 107 | AUSGABE | DEBUGGING-KONSOLE | **TERMINAL** | PORTS | KOMMENTARE

```
• * Task wird ausgeführt: docker compose -f "ws24-kp-avril/docker-compose.yml" up -d --build

WARN[0000] Found orphan containers ([pgadmin redis-server]) for this project. If you removed or r
enamed this service in your compose file, you can run this command with the --remove-orphans flag
to clean it up.
[+] Running 1/1
✓ Container postgres-db Started 0.4s

* Das Terminal wird von Aufgaben wiederverwendet, drücken Sie zum Schließen eine beliebige Tast
e.
[]
```

fish workspaces
Docker Com... ✓

> GLIEDERUNG
> ZEITACHSE
> DEPENDENCIES

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

2. Start Go Backend

The screenshot shows the Visual Studio Code interface with a workspace named 'ws24-kp-avril'. The Explorer sidebar on the right lists the project structure, including folders like '.devcontainer', '.vscode', 'backend-dart', 'backend-go', and 'docs', and files like '.env', '.gitattributes', '.gitignore', 'docker-compose.yml', and 'README.md'.

The main editor displays a README file with the following content:

```
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
```

The bottom panel shows the **TERMINAL** tab with the following output:

```
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
```

The terminal prompt is `vscode@4605c6876de3 /w/w/backend-go (main)>`.

3. Start Dart Backend

The screenshot shows the Visual Studio Code interface with a workspace named "ws24-kp-avril". The Explorer sidebar on the right lists the following files and folders:

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

The main editor displays the README for ActaTempus, which includes sections for Running ActaTempus, Backend (Go), Backend (Dart), and Database Management.

Running ActaTempus

Backend (Go)

```
cd backend-go
go run cmd/actatempus/main.go # Starts on port 8080
```

Backend (Dart)

```
cd backend-dart
dart run bin/backend_dart.dart # Starts on port 8080
```

Database Management

ActaTempus uses **Prisma ORM** for schema management and code generation across both backends. Start the PostgreSQL server using the provided `docker-compose.yml`:

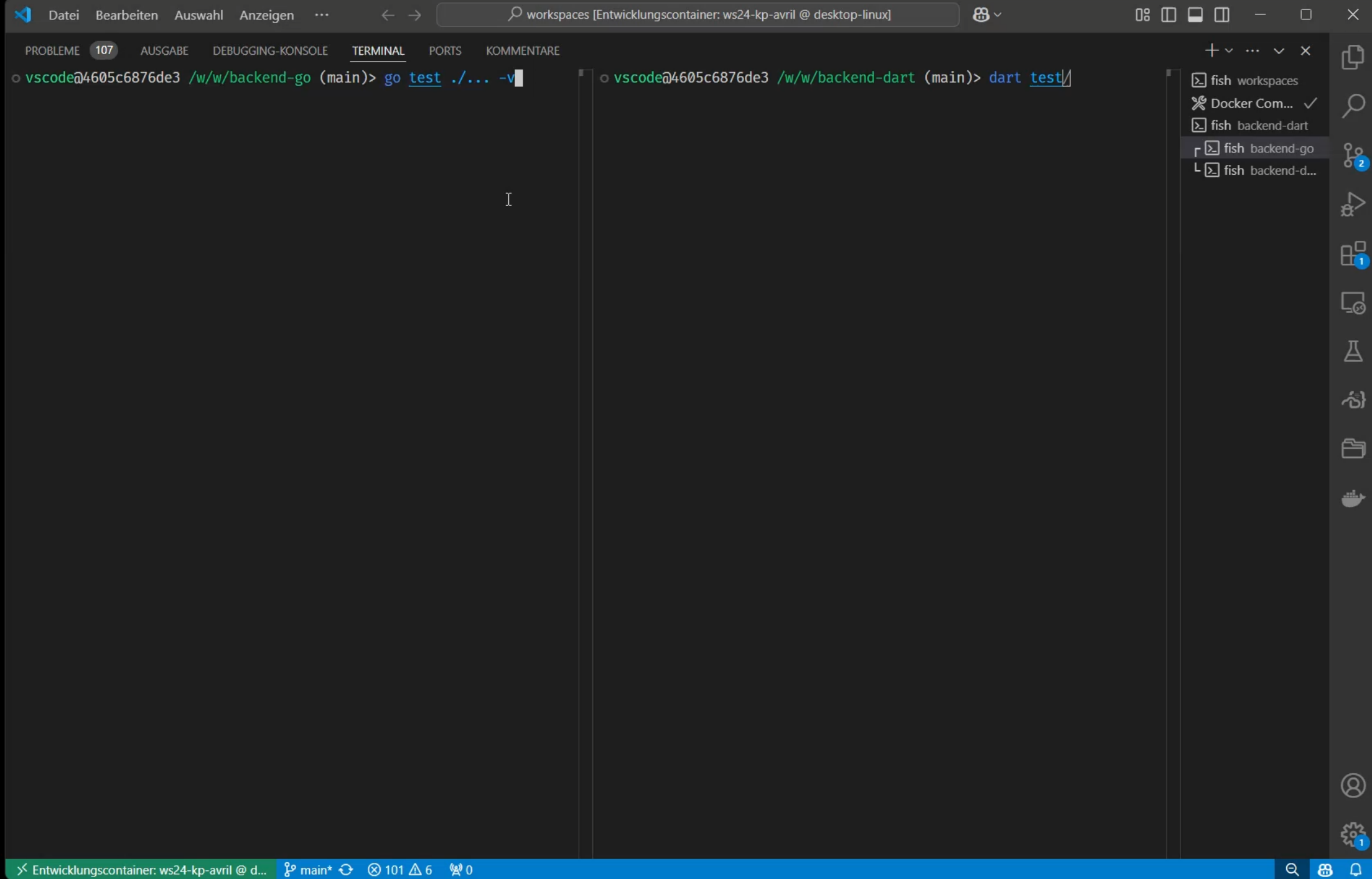
```
docker compose up
```

The terminal at the bottom shows the output of the `go run` command:

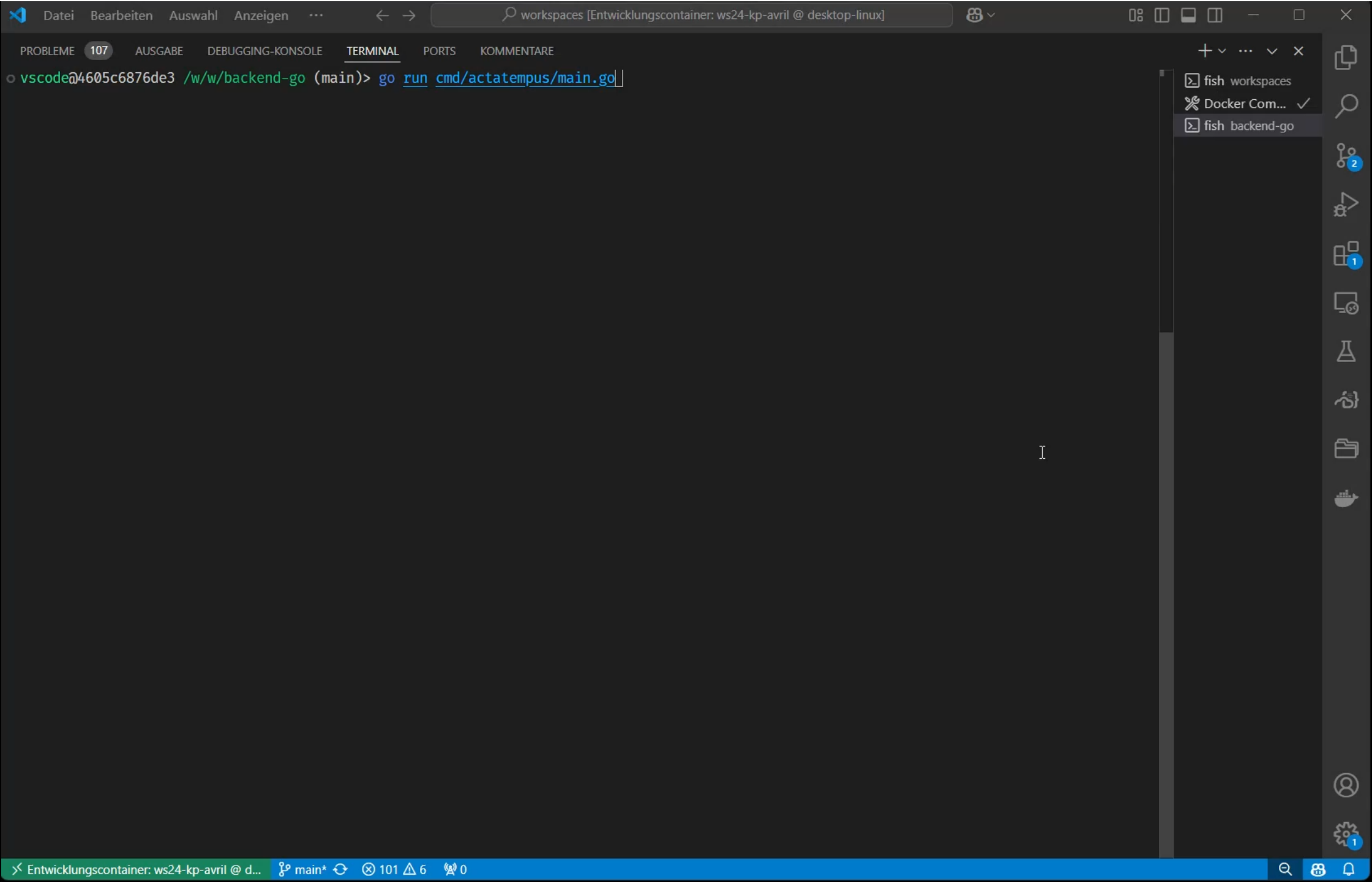
```
.(*AuthService).Login-fm (3 handlers)
[GIN-debug] POST    /api/auth/validate --> actatempus_backend/internal/application/services
.(*AuthService).Validate-fm (3 handlers)
[GIN-debug] POST    /api/auth/logout  --> actatempus_backend/internal/application/services
.(*AuthService).Logout-fm (3 handlers)
[GIN-debug] [WARNING] You trusted all proxies, this is NOT safe. We recommend you to set a value.
Please check https://pkg.go.dev/github.com/gin-gonic/gin#readme-don-t-trust-all-proxies for details.
[GIN-debug] Listening and serving HTTP on :8080
```

The status bar at the bottom indicates the current workspace is "Entwicklungscontainer: ws24-kp-avril @ d...", the active file is "main*", and there are 101 errors, 6 warnings, and 0 suggestions.

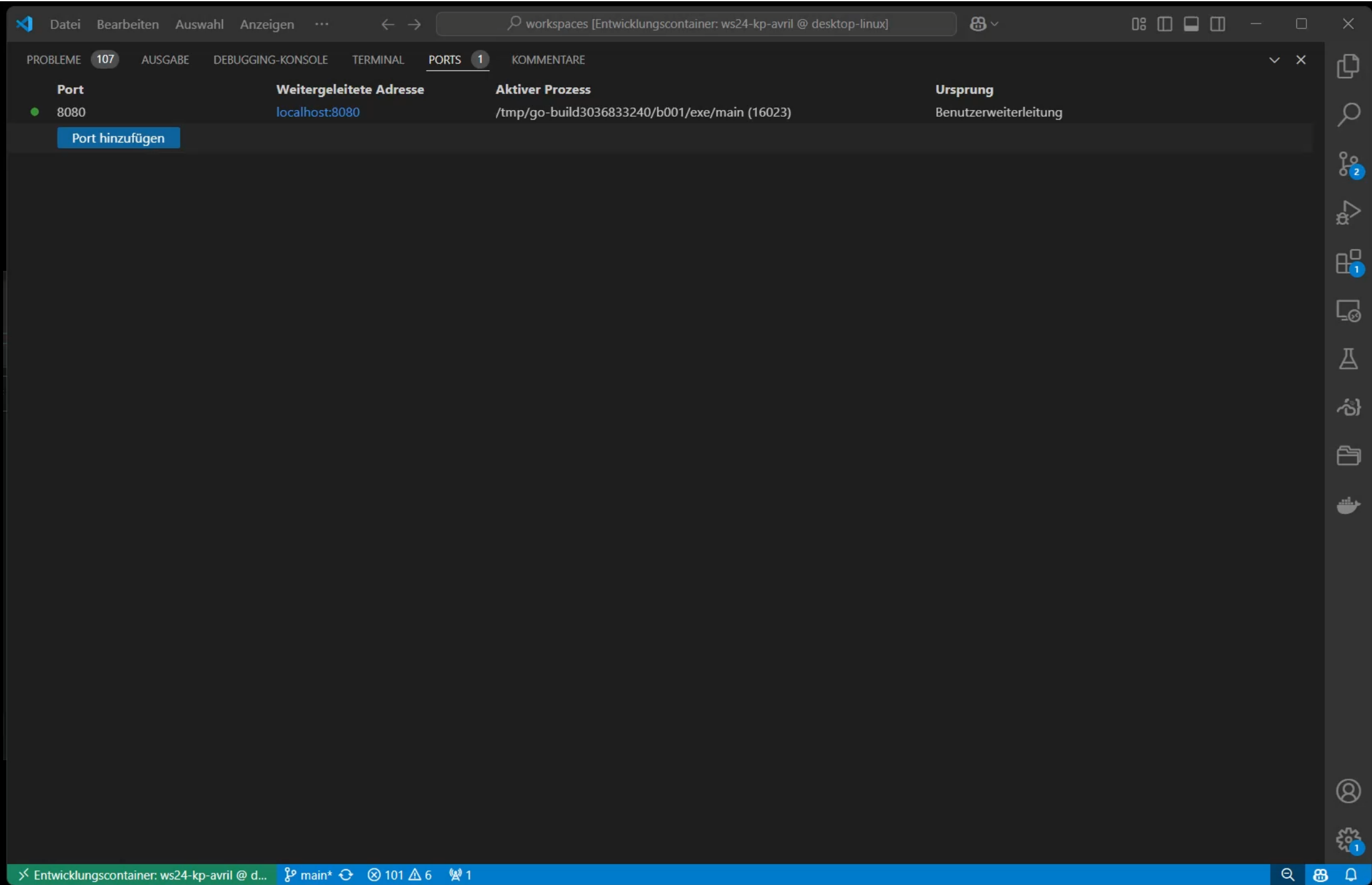
4. Tests



5. Demo: Starten



4.5. Demo: API-Client Setup (Insomnia)



4.5. Demo: Benutzer anlegen

The screenshot displays the Insomnia application interface. The top bar includes the application name, menu items (Application, File, Edit, View, Window, Tools, Help), a search bar, a star button, and a user count (35,444). The main workspace is divided into three panels:

- Left Panel (Scratch Pad):** Contains a list of environments (Base Environment, Add Cookies, Add Certificates) and a collection of requests categorized by folders: Auth (Login, Validate, Logout) and TimeEntries (GetAll, Get Time Entry By Id, Create Time Entry, Update Time Entry). The Auth folder is expanded, showing the 'Login' request with a POST method.
- Right Panel:** Displays a 'New Request' button and a 'New HTTP Request' button. Below these, there are buttons for 'Switch Requests' and 'Edit Environments'.
- Bottom Panel:** Shows the terminal output of a command, indicating that the application is running on port 8080.

The terminal output shows the following messages:

```
rs)
[GIN-debug] POST    /api/auth/validate --> actatempus_backend/internal/application/services.(*AuthService).Validate-fm (3 handlers)
[GIN-debug] POST    /api/auth/logout   --> actatempus_backend/internal/application/services.(*AuthService).Logout-fm (3 handlers)
[GIN-debug] [WARNING] You trusted all proxies, this is NOT safe. We recommend you to set a value.
Please check https://pkg.go.dev/github.com/gin-gonic/gin#readme-don-t-trust-all-proxies for details.
[GIN-debug] Listening and serving HTTP on :8080
```

The bottom status bar indicates the current environment is 'Entwicklungscontainer: ws24-kp-avril @ d...' and the main branch is 'main*'. The status bar also shows the number of open files (101), the number of errors (6), and the number of warnings (1).

4.5. Demo: Token verifizieren

Insomnia

Application File Edit View Window Tools Help

Search... Ctrl + P

Star 35.444

Login Sign up for free

Scratch Pad

Run

Base Environment

Manage Cookies (2)

Add Certificates

Filter

Auth

POST Login

POST Validate

POST Logout

TimeEntries

GET GetAll

GET Get Time Entry By Id

POST Create Time Entry

PUT Update Time Entry

ProjectTasks

GET Get All Tasks

GET Get Task By Id

POST http://localhost:8080/api/auth/login Send

200 OK 25 ms 229 B 2 Minutes Ago

Params Body Auth Headers 3 Scripts

Accept */*

Host <calculated at ru

Content-Type application/json

Preview Headers 5 Cookies 2 Tests 0/0 Mock

Preview

```
1 {
2   "token":
  "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE3Mz00OTQ3ODQsInVzZXI6IjRRCi6IjdjOTEzN2ExLTh1OWYtNGY4OC04NzZkLTlYxNDk1YWZlYzg5MSJ9.W3Wu6zNHhRJs1vYEKqF-gaxQaNkX2D83CpSVoSsdF7w",
3   "userId": "7c9137a1-8b9f-4f88-876d-61495afec891"
4 }
```

Bulk Edit \$.store.books[*].author

Preferences We have detected orphaned projects on your computer, click here to view them.

Log in to see your projects Made with by Kong

ers)

[GIN-debug] [WARNING] You trusted all proxies, this is NOT safe. We recommend you to set a value. Please check https://pkg.go.dev/github.com/gin-gonic/gin#readme-don-t-trust-all-proxies for details.

[GIN-debug] Listening and serving HTTP on :8080

[GIN] 2025/01/19 - 17:39:10 | 401 | 24.546915ms | 127.0.0.1 | GET | "/api/users/"

[GIN] 2025/01/19 - 17:39:30 | 201 | 16.712502ms | 127.0.0.1 | POST | "/api/users/"

[GIN] 2025/01/19 - 17:39:44 | 200 | 5.393972ms | 127.0.0.1 | POST | "/api/auth/login"

[GIN] 2025/01/19 - 17:40:54 | 200 | 52.691µs | 127.0.0.1 | POST | "/api/auth/validate"

Entwicklungscontainer: ws24-kp-avril @ d... main* 101 6 1

4.5. Demo: Zeiteinträge anlegen

The screenshot displays the Insomnia API client interface. The left sidebar shows a project named 'Scratch Pad' with a list of endpoints under the 'Auth' folder: Login (POST), Validate (POST), Logout (POST), GetAll (GET), Get Time Entry By Id (GET), Create Time Entry (POST), Update Time Entry (PUT), Get All Tasks (GET), and Get Task By Id (GET). The 'Validate' endpoint is selected, showing a POST request to 'http://localhost:8080/api/auth/validate'. The response is a 200 OK status with a response time of 13 ms and a body size of 78 B. The response body is a JSON object: { "message": "Token validated", "user_id": "7c9137a1-8b9f-4f88-876d-61495afec891" }. The bottom panel shows a log of HTTP requests and responses, including a warning about trusted proxies and a list of requests with their status codes, response times, and URLs.

Insomnia

Application File Edit View Window Tools Help

Search.. Ctrl + P Star 35,444 Login Sign up for free

Scratch Pad Run

Base Environment Manage Cookies (2) Add Certificates

Filter

Auth

- POST Login
- POST Validate
- POST Logout

TimeEntries

- GET GetAll
- GET Get Time Entry By Id
- POST Create Time Entry
- PUT Update Time Entry

ProjectTasks

- GET Get All Tasks
- GET Get Task By Id

POST http://localhost:8080/api/auth/validate Send 200 OK 13 ms 78 B 2 Minutes Ago

Params Body Auth Headers (2) Scripts Docs Preview Headers (3) Cookies Tests 0/0 → Mock

Accept */*

Host <calculated at ru

header value

Preview

```
1 {
2   "message": "Token validated",
3   "user_id": "7c9137a1-8b9f-4f88-876d-61495afec891"
4 }
```

\$.store.books[*].author

Preferences We have detected orphaned projects on your computer, click here to view them. Log in to see your projects Made with ♥ by Kong

[GIN-debug] [WARNING] You trusted all proxies, this is NOT safe. We recommend you to set a value. Please check https://pkg.go.dev/github.com/gin-gonic/gin#readme-don-t-trust-all-proxies for details.

[GIN-debug] Listening and serving HTTP on :8080

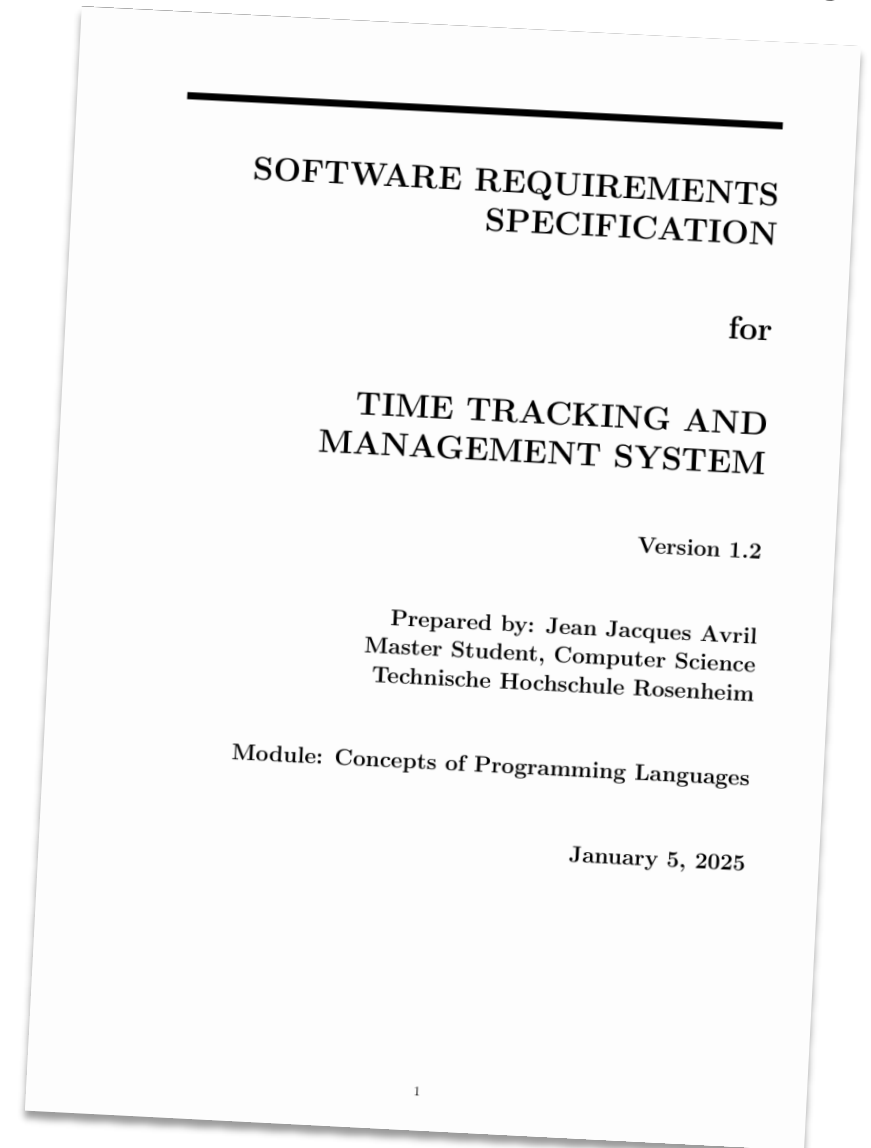
[GIN]	2025/01/19 - 17:39:10	401	24.546915ms	127.0.0.1	GET	"/api/users/"
[GIN]	2025/01/19 - 17:39:30	201	16.712502ms	127.0.0.1	POST	"/api/users/"
[GIN]	2025/01/19 - 17:39:44	200	5.393972ms	127.0.0.1	POST	"/api/auth/login"
[GIN]	2025/01/19 - 17:40:54	200	52.691µs	127.0.0.1	POST	"/api/auth/validate"
[GIN]	2025/01/19 - 17:42:32	200	45.101µs	127.0.0.1	POST	"/api/auth/validate"

Entwicklungscontainer: ws24-kp-avril @ d... main* 101 6 1

Umsetzung

Spezifikation

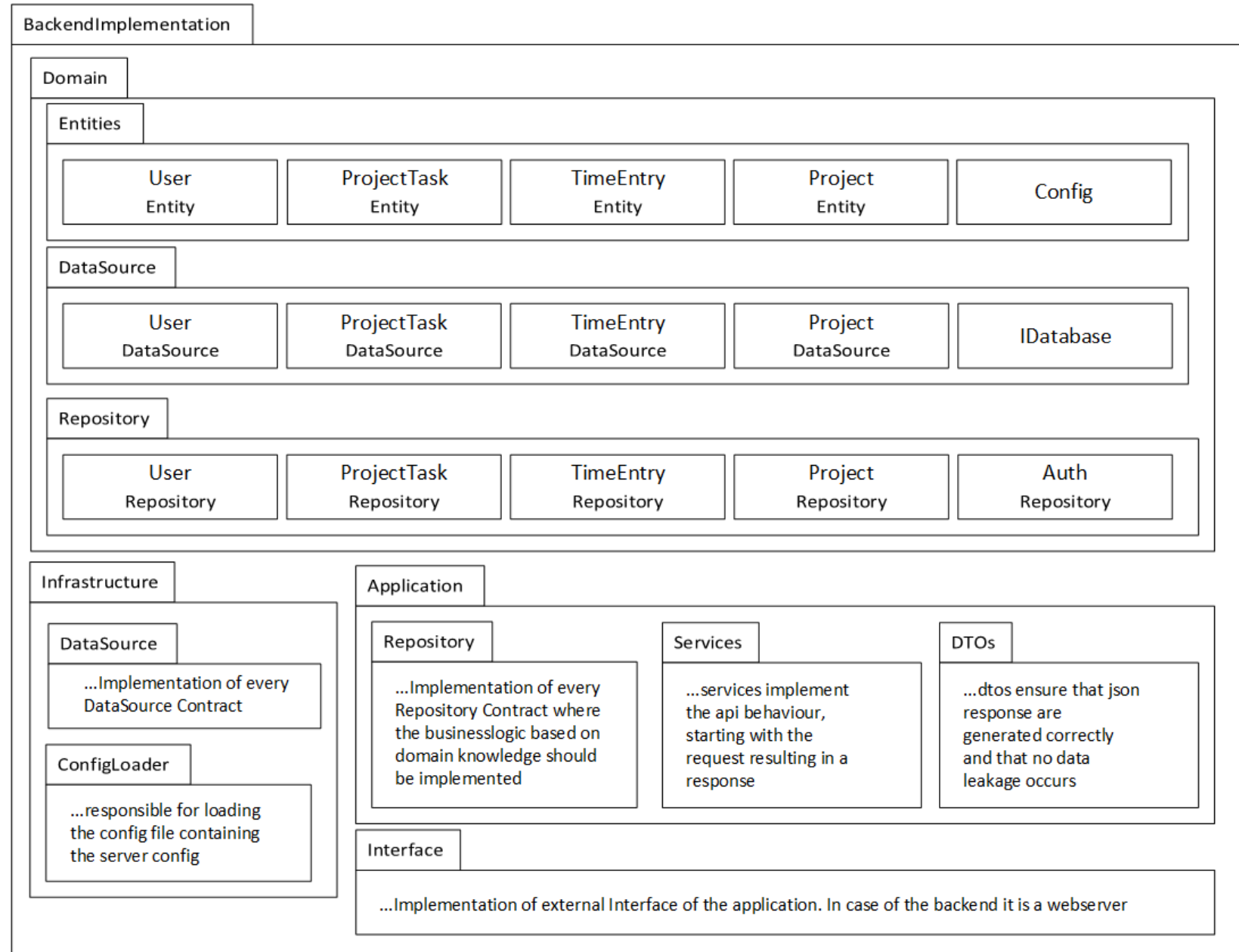
1. Einführung: Ziel & Scope
2. Anforderungen
3. Technologie und Domänenwissen
4. System Design: Architektur, Module, Datenbank
5. Implementierungsdetails Go Backend
6. Implementierungsdetails Dart Backend
7. Definition der API-Endpunkte



DDD Architektur

Umsetzung

Domain Driven Design



FP Sprachfeatures und Eignung

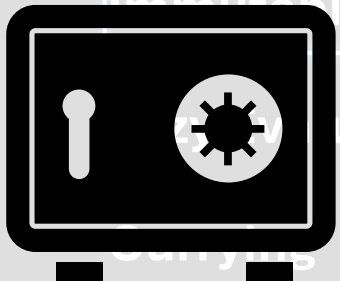
Benötigtes FP-Feature	Dart	Go
Erstklassige Funktionen	✓ Unterstützt	✓ Unterstützt
High-Order Functions	✓ Unterstützt	✓ Unterstützt
Closures	✓ Unterstützt	✓ Unterstützt
Generics	✓ Unterstützt (Null Safety integriert)	✓ Unterstützt (ab Go 1.18)
Statische Typisierung	✓ Unterstützt	✓ Unterstützt
Nebenläufigkeit	✓ 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



Dart Immutability-Beispiel

```
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]
```

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

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

Go

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

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

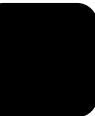
Immutability mit Freezed

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

```
@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

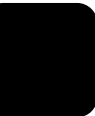
```
Function(A) Function(B) curry<A, B, R>(R Function(A, B) f) {  
    return (A a) ⇒ (B b) ⇒ f(a, b);  
}  
  
void main() {  
    int add(int x, int y) ⇒ x + y;  
  
    final curriedAdd = curry(add);  
  
    final add5 = curriedAdd(5);  
  
    final result = add5(3);  
  
    print(result); // 8  
}
```

● ● ● Go Currying-Beispiel

```
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

Einführung in fpdart

fpdart ist ein Paket, das funktionale Programmierkonzepte nach Dart bringt.



Task/TaskEither

Asynchrone
Programmierung



Either

Funktionale
Fehlerbehandlung



Monaden

Elegantes
Verkettungs-
Handling



Option

Umgang mit
optionalen Werten

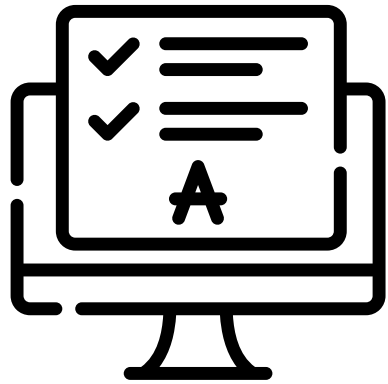
```
@Route.get('/')
Future<Response> listEntries(Request request) async {
  return checkAuth(request, authRepository) // TaskEither<IError, String>
    .flatMap(
      (_) => entries.findAll() // TaskEither<IError, List<TimeEntry>>
    )
    .flatMap(mapper.listTo) // TaskEither<IError, List<TimeEntryDto>>
    .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

```
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



Überraschend 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