Un linguaggio per domarli tutti
Dart full-stack con Serverpod

Dart & Flutter dal client

al server in perfetta armonia

Federico ParezzanX GDG Vicenza

14.06.2025

Federico Parezzan

Senior Software Engineer

Java Oracle Certified

Flutter Developer

Agenda

Agenda

Agenda

Agenda

Agenda

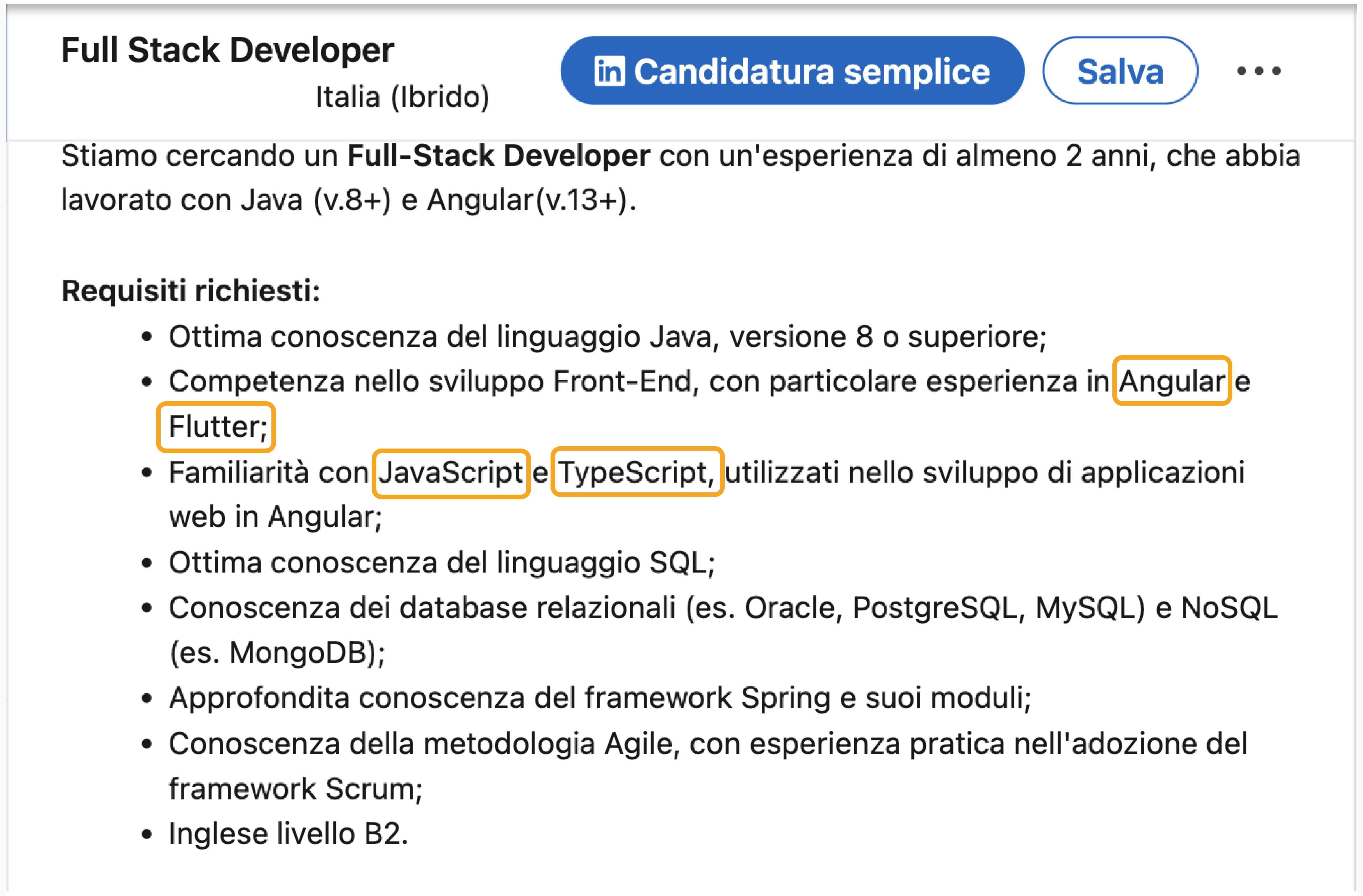
Agenda

Why Dart full-stack?

Full-stack… with how many stacks?

Frontend

Angular
Flutter
Javascript
Typescript



Full Stack Developer
Italia (Ibrido)

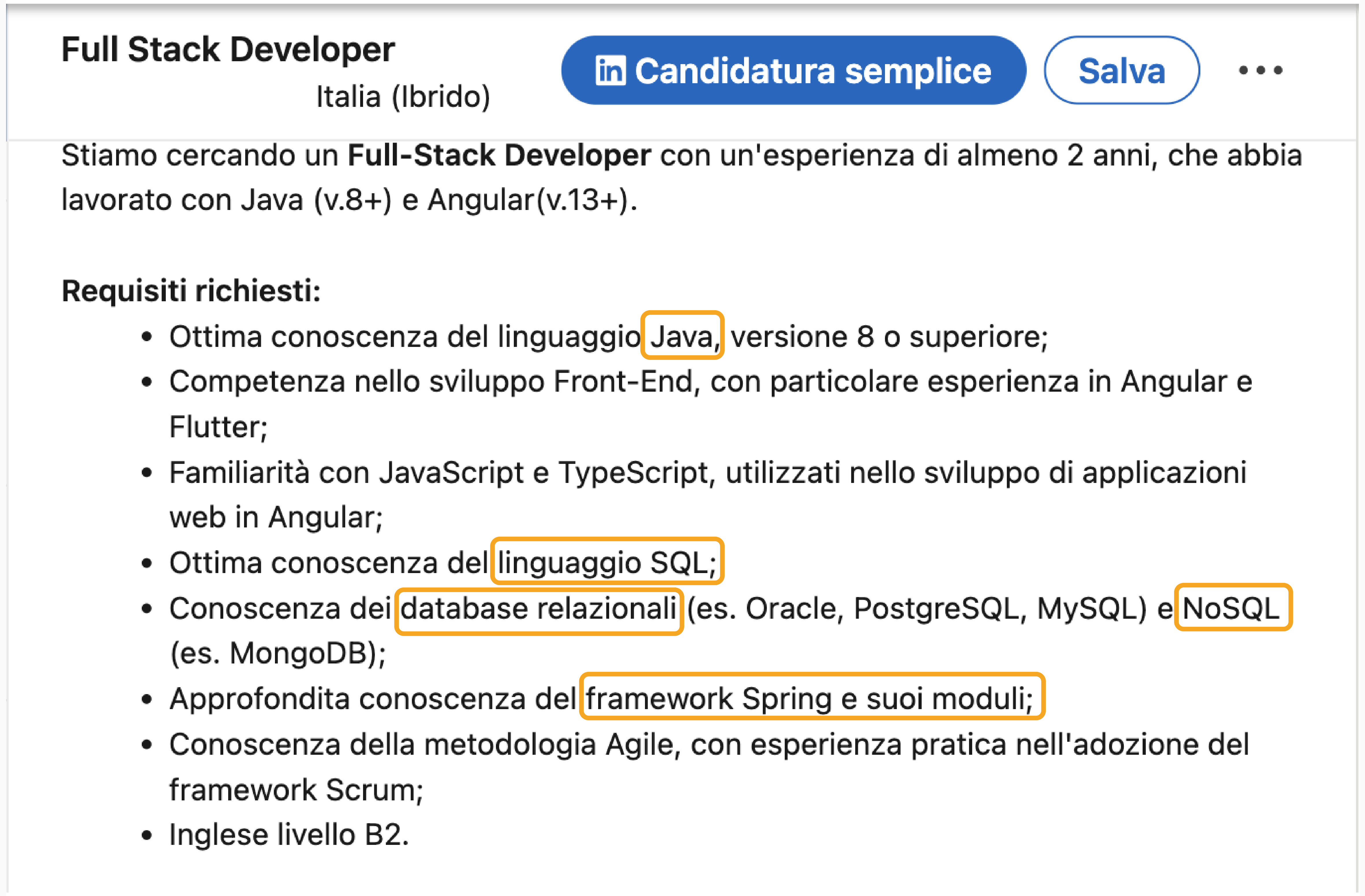
Stiamo cercando un **Full-Stack Developer** con un'esperienza di almeno 2 anni, che abbia lavorato con Java (v.8+) e Angular(v.13+).

Requisiti richiesti:

- Ottima conoscenza del linguaggio Java, versione 8 o superiore;
- Competenza nello sviluppo Front-End, con particolare esperienza in Angular e Flutter;
- Familiarità con JavaScript e TypeScript, utilizzati nello sviluppo di applicazioni web in Angular;
- Ottima conoscenza del linguaggio SQL;
- Conoscenza dei database relazionali (es. Oracle, PostgreSQL, MySQL) e NoSQL (es. MongoDB);
- Approfondita conoscenza del framework Spring e suoi moduli;
- Conoscenza della metodologia Agile, con esperienza pratica nell'adozione del framework Scrum;
- Inglese livello B2.

Backend

Java
Sql
Relational Databases
NoSql database
Spring framework



Full Stack Developer
Italia (Ibrido)

Stiamo cercando un **Full-Stack Developer** con un'esperienza di almeno 2 anni, che abbia lavorato con Java (v.8+) e Angular(v.13+).

Requisiti richiesti:

- Ottima conoscenza del linguaggio Java, versione 8 o superiore;
- Competenza nello sviluppo Front-End, con particolare esperienza in Angular e Flutter;
- Familiarità con JavaScript e TypeScript, utilizzati nello sviluppo di applicazioni web in Angular;
- Ottima conoscenza del linguaggio SQL;
- Conoscenza dei database relazionali (es. Oracle, PostgreSQL, MySQL) e NoSQL (es. MongoDB);
- Approfondita conoscenza del framework Spring e suoi moduli;
- Conoscenza della metodologia Agile, con esperienza pratica nell'adozione del framework Scrum;
- Inglese livello B2.

Full-stack... in Dart?

WHY DART FULL-STACK?

Frontend in Flutter

Backend requires many technologies

Dart used server side

One codebase, same language, shared models

Introducing Serverpod

Serverpod is an open-source, scalable app server, written in Dart for the Flutter community.

Scalable & progressive

Modular

Scalable & progressive

Modular

Scalable

Scalable & progressive

Modular

Scalable

Flexible

Benefits

Reduced complexity

Benefits

Reduced complexity

Open and free

Benefits

Reduced
complexity

Open
and free

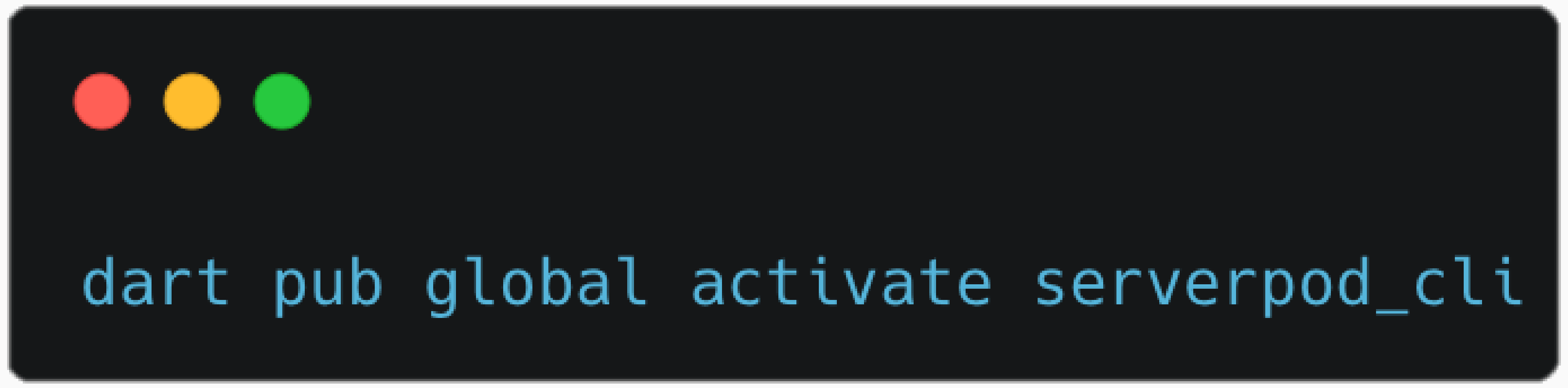
Stable and reliable

Key features

- ORM
- Real-time capabilities
- Straightforward authentication
- All essentials covered
- Cloud ready

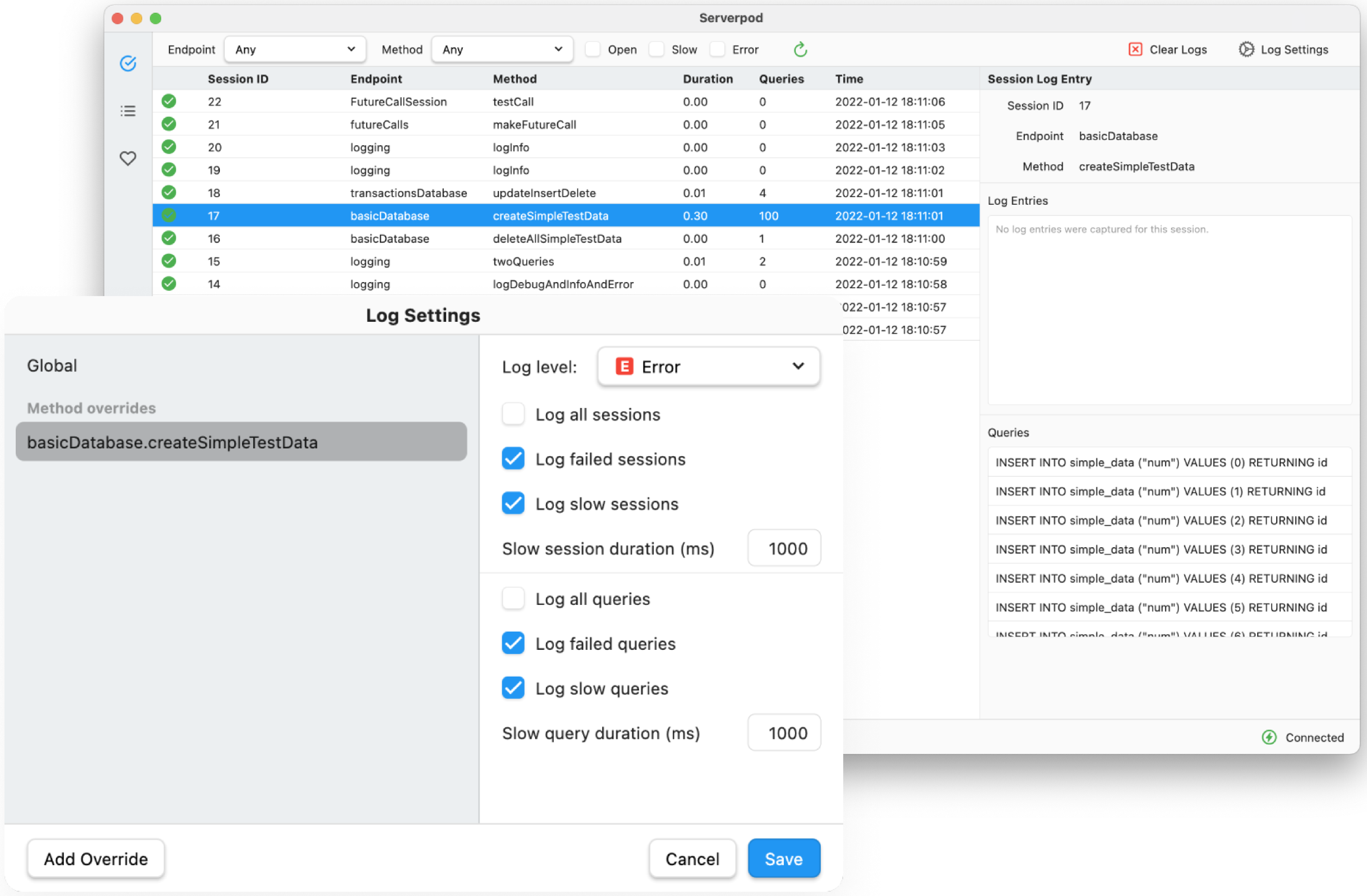
Architecture & Project Setup

Install CLI



```
dart pub global activate serverpod_cli
```

Serverpod Insights



Serverpod

Endpoint Any · Method Any · Open · Slow · Error · Clear Logs · Log Settings

Session ID	Endpoint	Method	Duration	Queries	Time
22	FutureCallSession	testCall	0.00	0	2022-01-12 18:11:06
21	futureCalls	makeFutureCall	0.00	0	2022-01-12 18:11:05
20	logging	logInfo	0.00	0	2022-01-12 18:11:03
19	logging	logInfo	0.00	0	2022-01-12 18:11:02
18	transactionsDatabase	updateInsertDelete	0.01	4	2022-01-12 18:11:01
17	basicDatabase	createSimpleTestData	0.30	100	2022-01-12 18:11:01
16	basicDatabase	deleteAllSimpleTestData	0.00	1	2022-01-12 18:11:00
15	logging	twoQueries	0.01	2	2022-01-12 18:10:59
14	logging	logDebugAndInfoAndError	0.00	0	2022-01-12 18:10:58
					022-01-12 18:10:57
					022-01-12 18:10:57

Session Log Entry

Session ID 17
Endpoint basicDatabase
Method createSimpleTestData

Log Entries

No log entries were captured for this session.

Queries

```
INSERT INTO simple_data ("num") VALUES (0) RETURNING id
INSERT INTO simple_data ("num") VALUES (1) RETURNING id
INSERT INTO simple_data ("num") VALUES (2) RETURNING id
INSERT INTO simple_data ("num") VALUES (3) RETURNING id
INSERT INTO simple_data ("num") VALUES (4) RETURNING id
INSERT INTO simple_data ("num") VALUES (5) RETURNING id
```

Connected

Log Settings

Global

Method overrides

basicDatabase.createSimpleTestData

Log level: Error

- [ ] Log all sessions
- [x] Log failed sessions
- [x] Log slow sessions

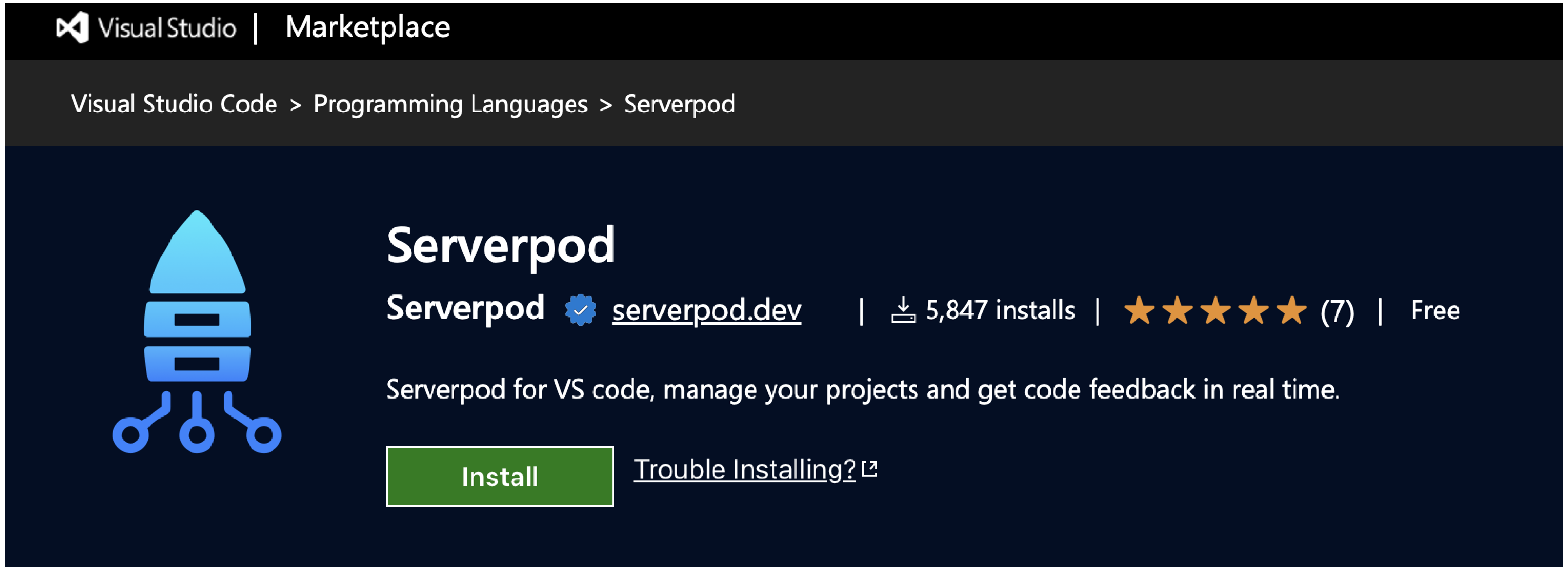
Slow session duration (ms) 1000

- [ ] Log all queries
- [x] Log failed queries
- [x] Log slow queries

Slow query duration (ms) 1000

Add Override · Cancel · Save

VS Code extension


Visual Studio | Marketplace
Visual Studio Code > Programming Languages > Serverpod

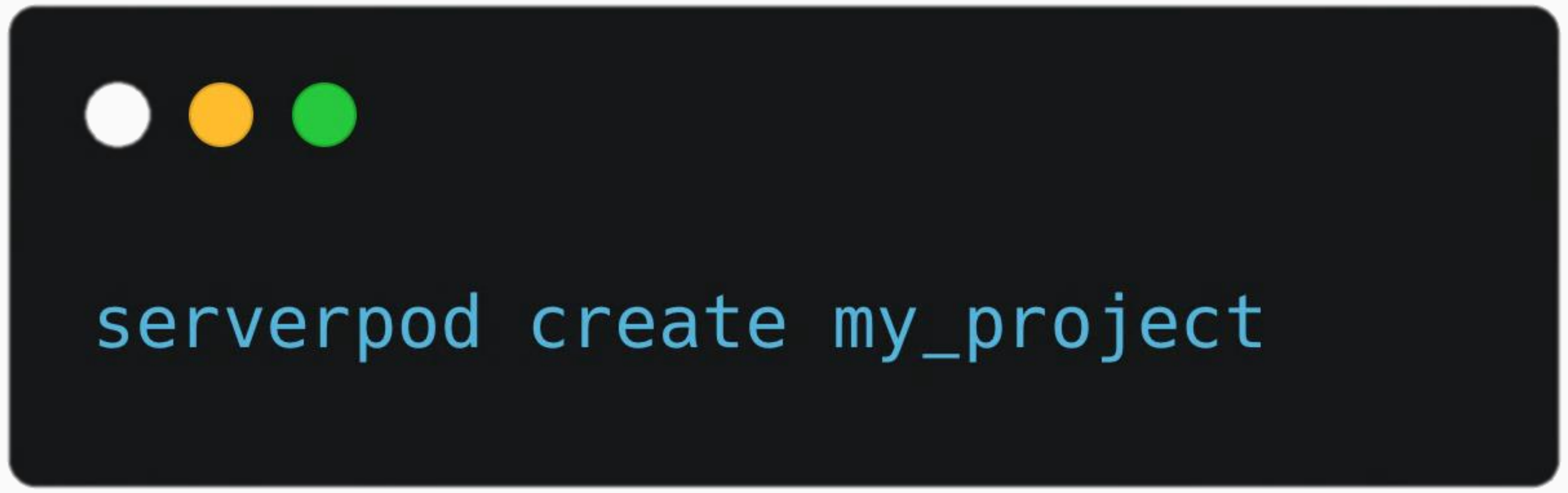
Serverpod

Serverpod serverpod.dev | 5,847 installs | ★★★★★ (7) | Free

Serverpod for VS code, manage your projects and get code feedback in real time.

Install
Trouble Installing?

Create project



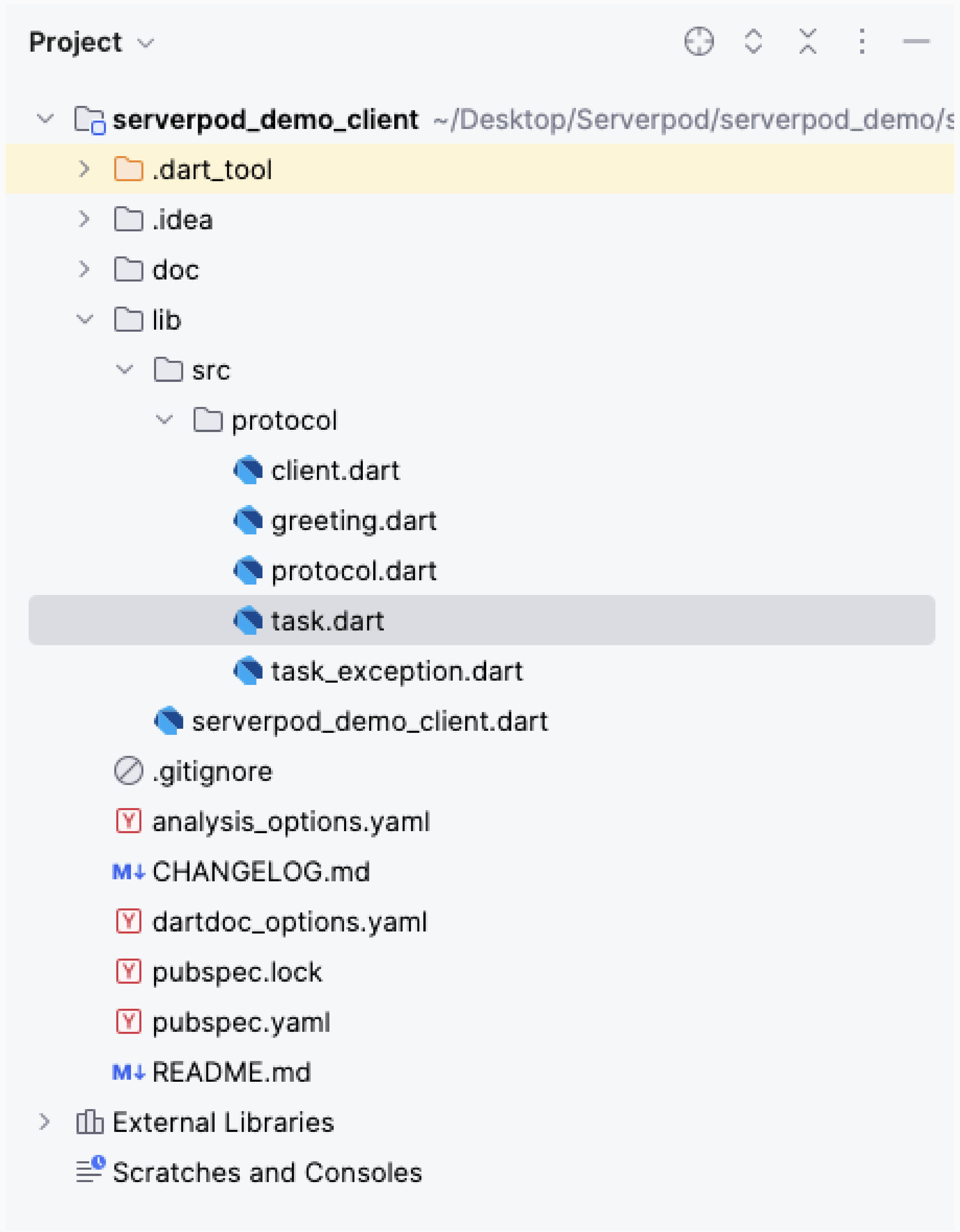
```
serverpod create my_project
```

Serverpod requires a PostgreSQL database

Serverpod pj provides a *docker-compose.yaml*

API client → xxx_client

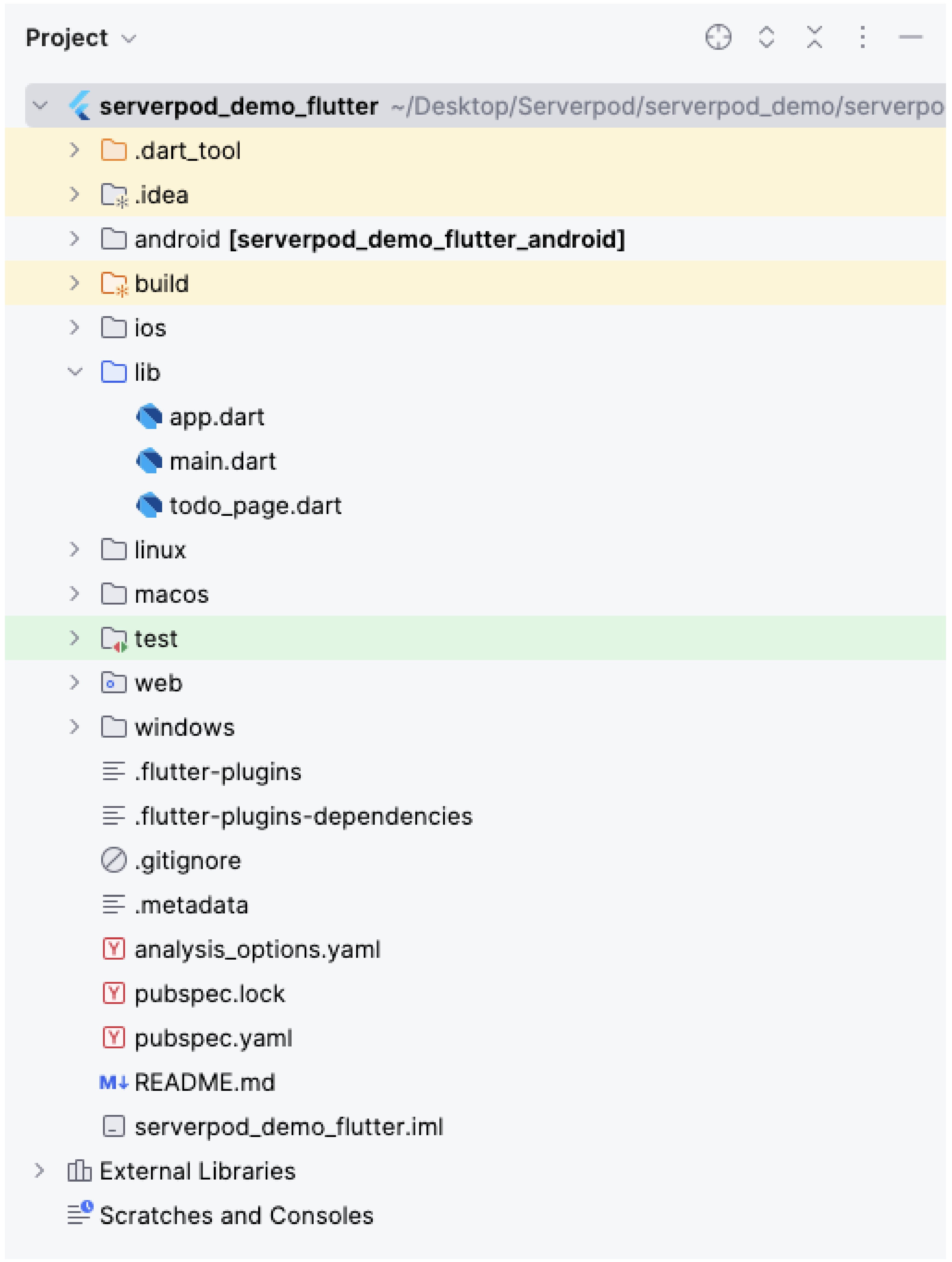
Stores the program code generated from the Server side.
Allows the APP side to access it.
Bridge for communication between the two ends.



```
Project
serverpod_demo_client ~/Desktop/Serverpod/serverpod_demo/s
  .dart_tool
  .idea
  doc
  lib
    src
      protocol
        client.dart
        greeting.dart
        protocol.dart
        task.dart
        task_exception.dart
    serverpod_demo_client.dart
  .gitignore
  analysis_options.yaml
  CHANGELOG.md
  dartdoc_options.yaml
  pubspec.lock
  pubspec.yaml
  README.md
External Libraries
Scratches and Consoles
```

Flutter App → xxx_flutter

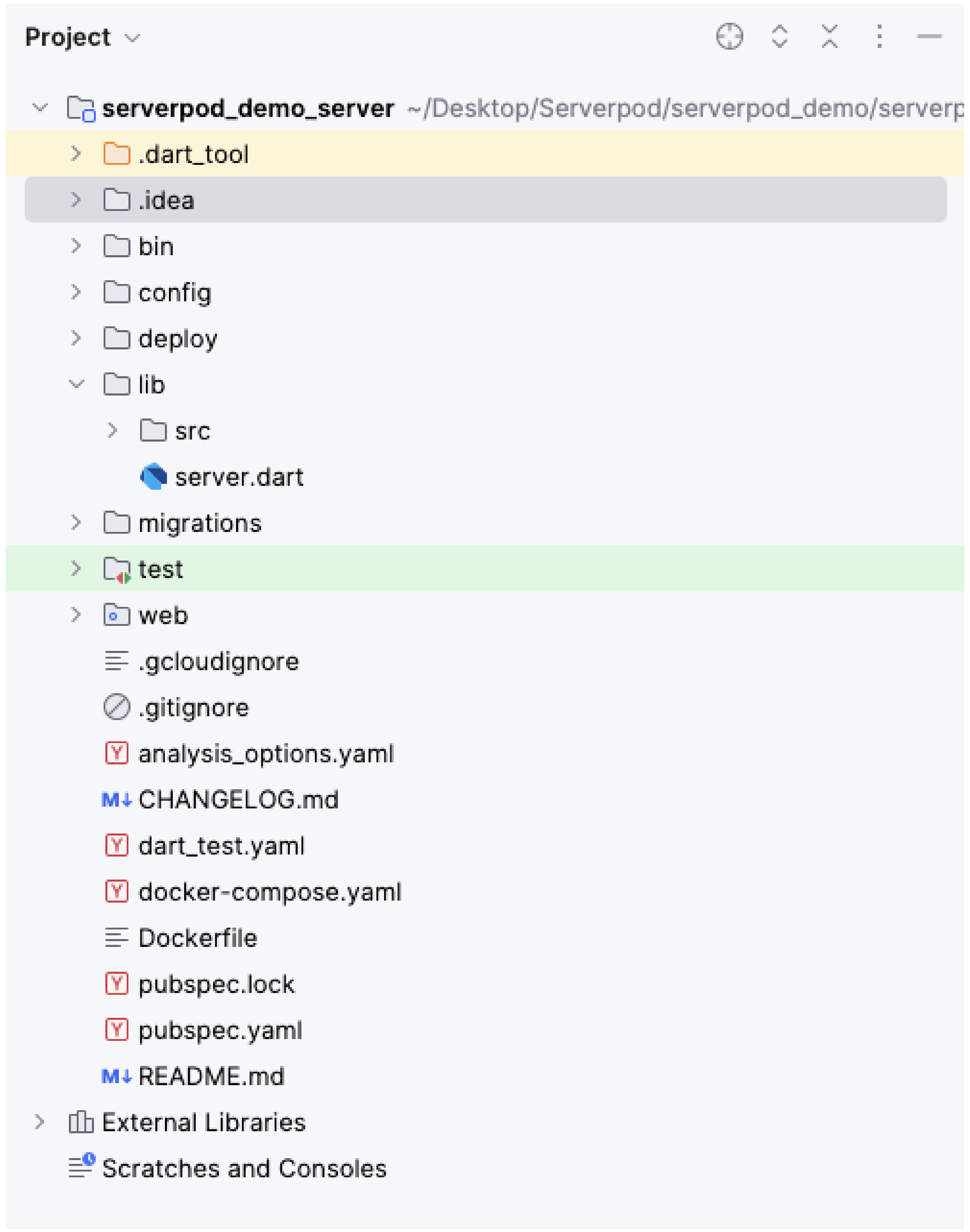
The frontend application (mobile/desktop/web)


Project

```
serverpod_demo_flutter ~/Desktop/Serverpod/serverpod_demo/serverpo
  .dart_tool
  .idea
  android [serverpod_demo_flutter_android]
  build
  ios
  lib
    app.dart
    main.dart
    todo_page.dart
  linux
  macos
  test
  web
  windows
  .flutter-plugins
  .flutter-plugins-dependencies
  .gitignore
  .metadata
  analysis_options.yaml
  pubspec.lock
  pubspec.yaml
  README.md
  serverpod_demo_flutter.iml
External Libraries
Scratches and Consoles
```

Server App → xxx_server

The backend application



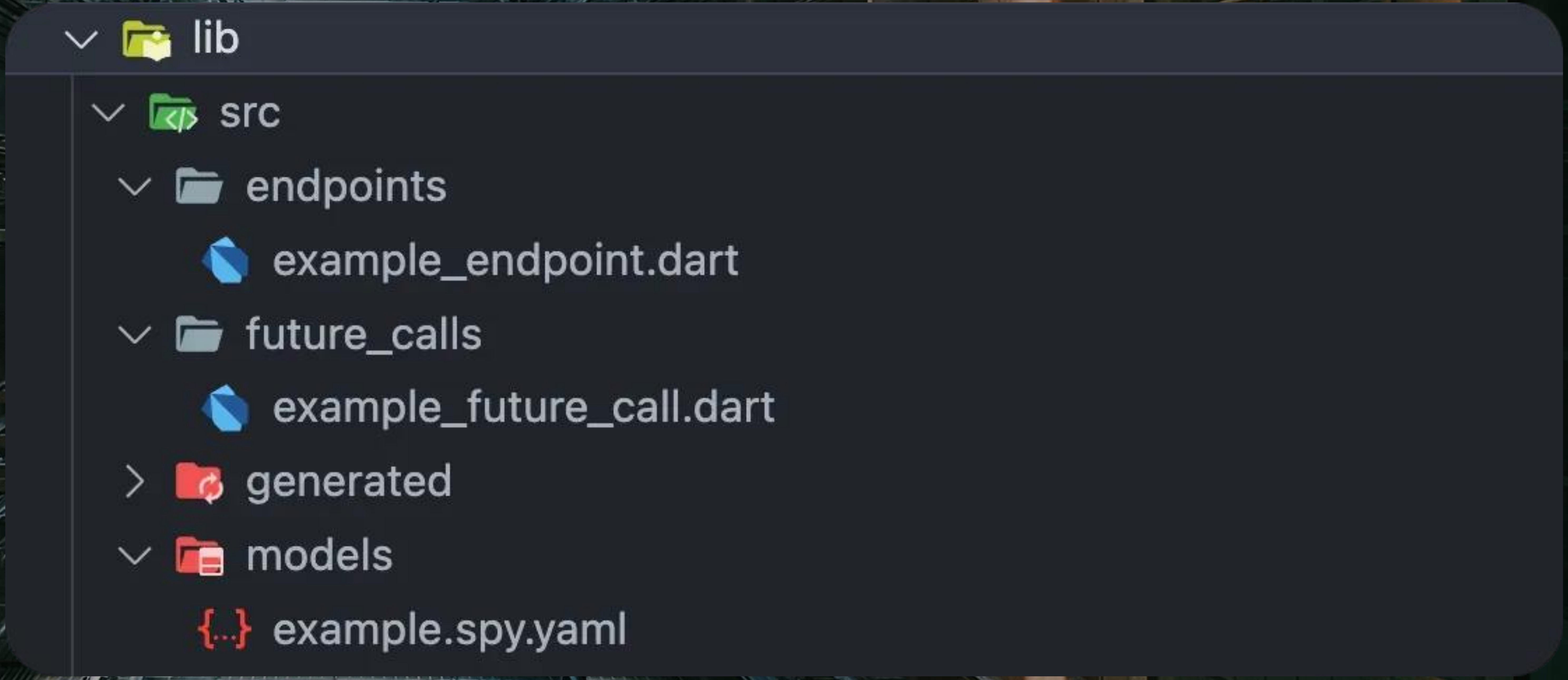
Project

```
v serverpod_demo_server ~/Desktop/Serverpod/serverpod_demo/serverp
  > .dart_tool
  > .idea
  > bin
  > config
  > deploy
  v lib
    > src
      server.dart
  > migrations
  > test
  > web
    .gcloudignore
    .gitignore
    analysis_options.yaml
    CHANGELOG.md
    dart_test.yaml
    docker-compose.yaml
    Dockerfile
    pubspec.lock
    pubspec.yaml
    README.md
> External Libraries
  Scratches and Consoles
```

Server directories

01

endpoints/

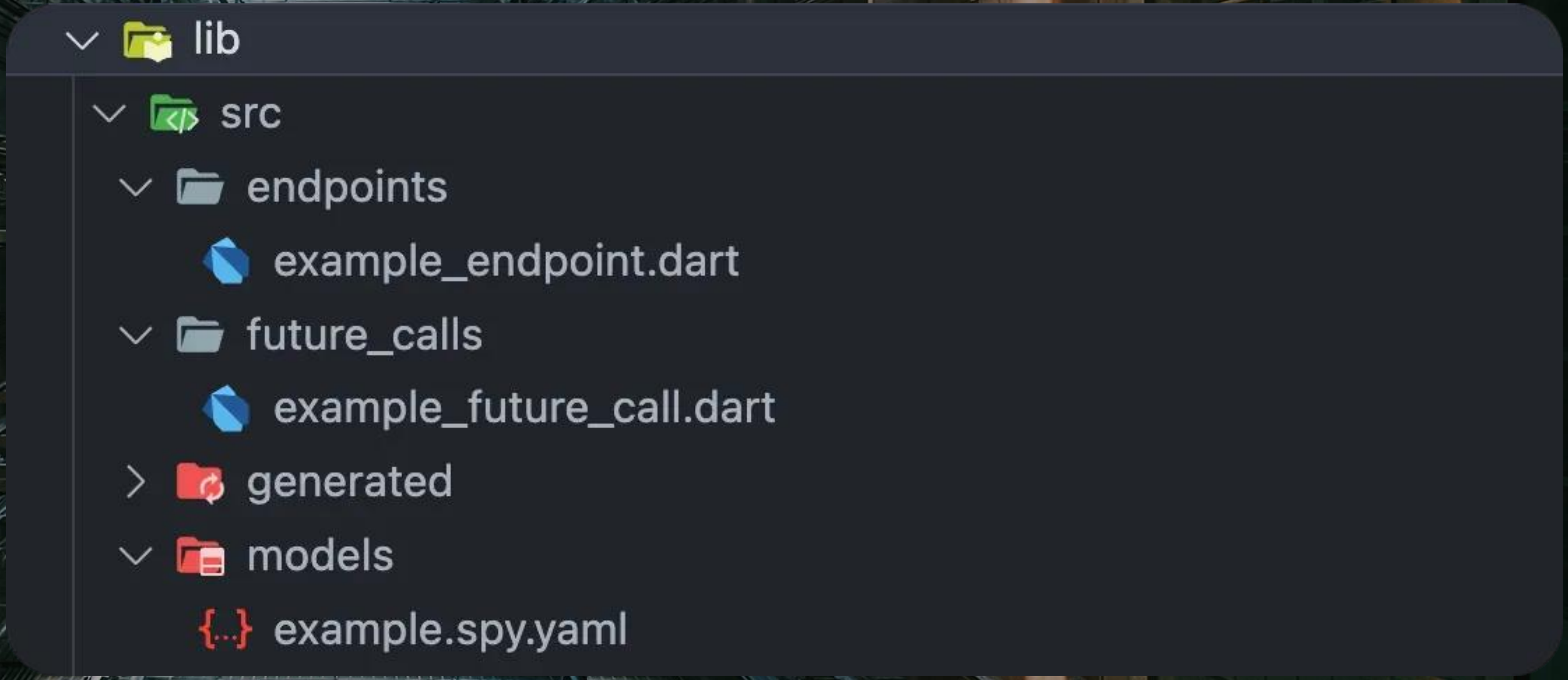


```
lib
  src
    endpoints
      example_endpoint.dart
    future_calls
      example_future_call.dart
    generated
    models
      example.spy.yaml
```

Server directories

02

models/

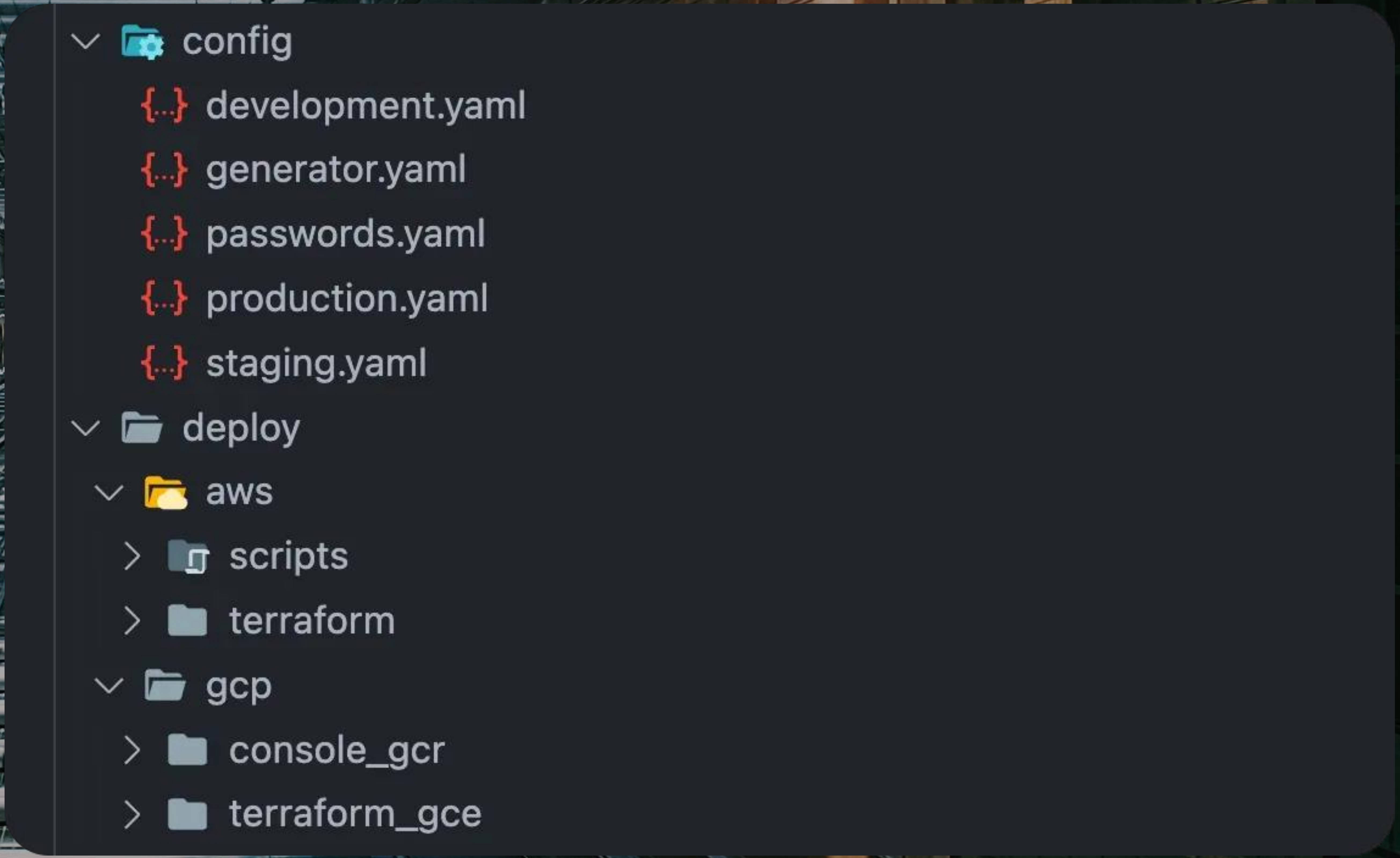


```
lib
  src
    endpoints
      example_endpoint.dart
    future_calls
      example_future_call.dart
    generated
    models
      example.spy.yaml
```

Server directories

03

config/

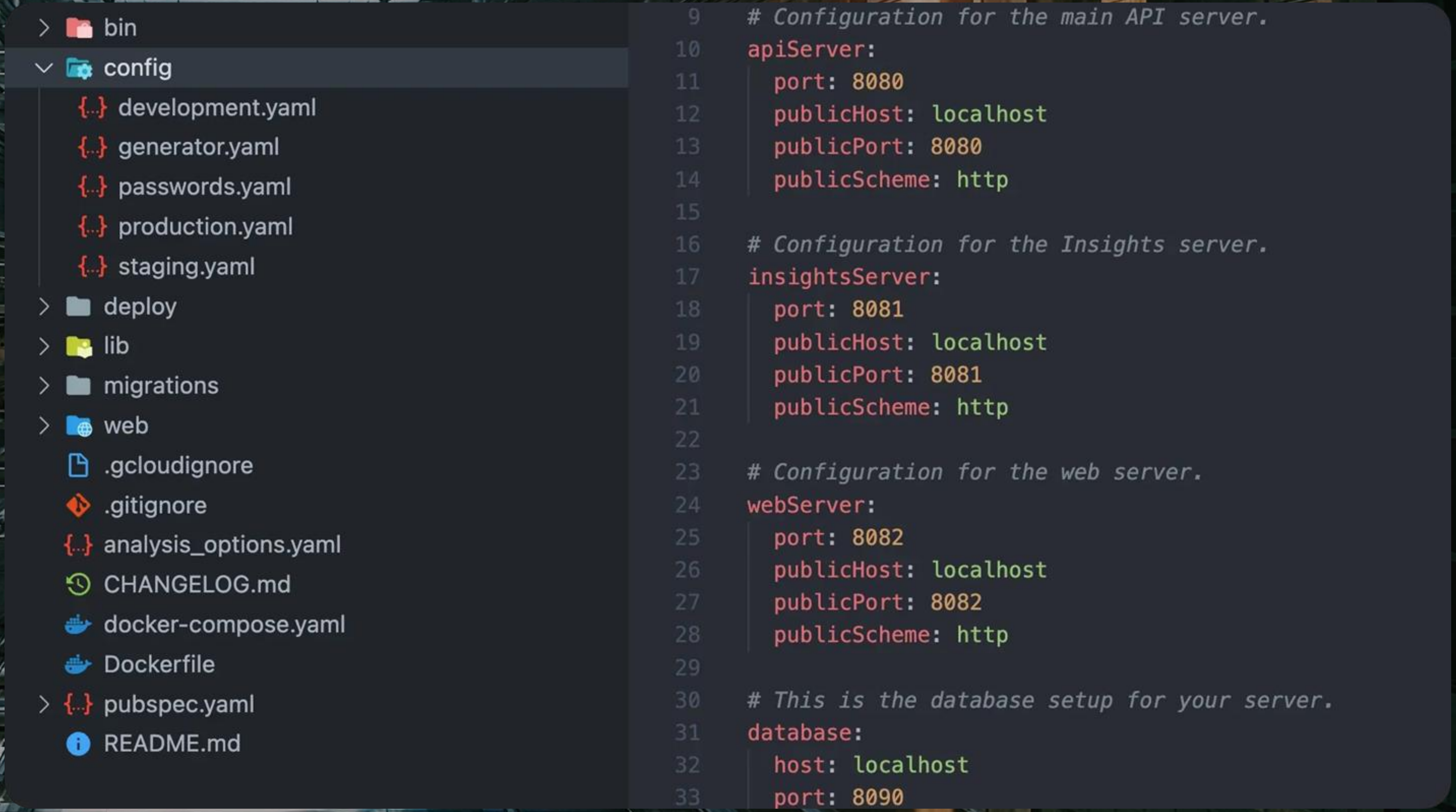


```
⌄ config
    development.yaml
    generator.yaml
    passwords.yaml
    production.yaml
    staging.yaml
⌄ deploy
  ⌄ aws
    › scripts
    › terraform
  ⌄ gcp
    › console_gcr
    › terraform_gce
```

Server directories

03

config/



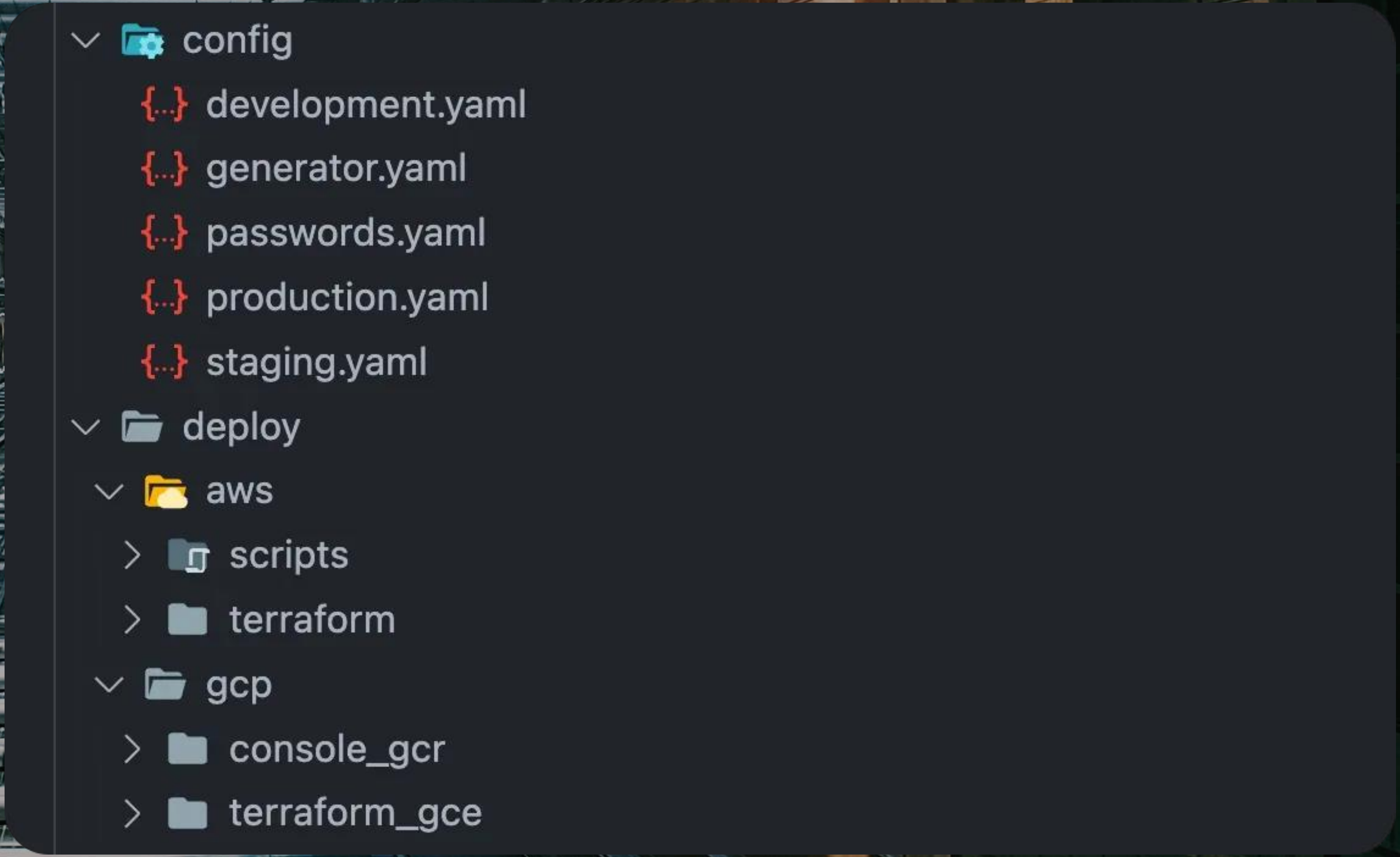
```
> bin
v config
    development.yaml
    generator.yaml
    passwords.yaml
    production.yaml
    staging.yaml
> deploy
> lib
> migrations
> web
  .gcloudignore
  .gitignore
  analysis_options.yaml
  CHANGELOG.md
  docker-compose.yaml
  Dockerfile
> pubspec.yaml
  README.md
```

```yaml
# Configuration for the main API server.
apiServer:
  port: 8080
  publicHost: localhost
  publicPort: 8080
  publicScheme: http

# Configuration for the Insights server.
insightsServer:
  port: 8081
  publicHost: localhost
  publicPort: 8081
  publicScheme: http

# Configuration for the web server.
webServer:
  port: 8082
  publicHost: localhost
  publicPort: 8082
  publicScheme: http

# This is the database setup for your server.
database:
  host: localhost
  port: 8090
```

Server directories

04

deploy/

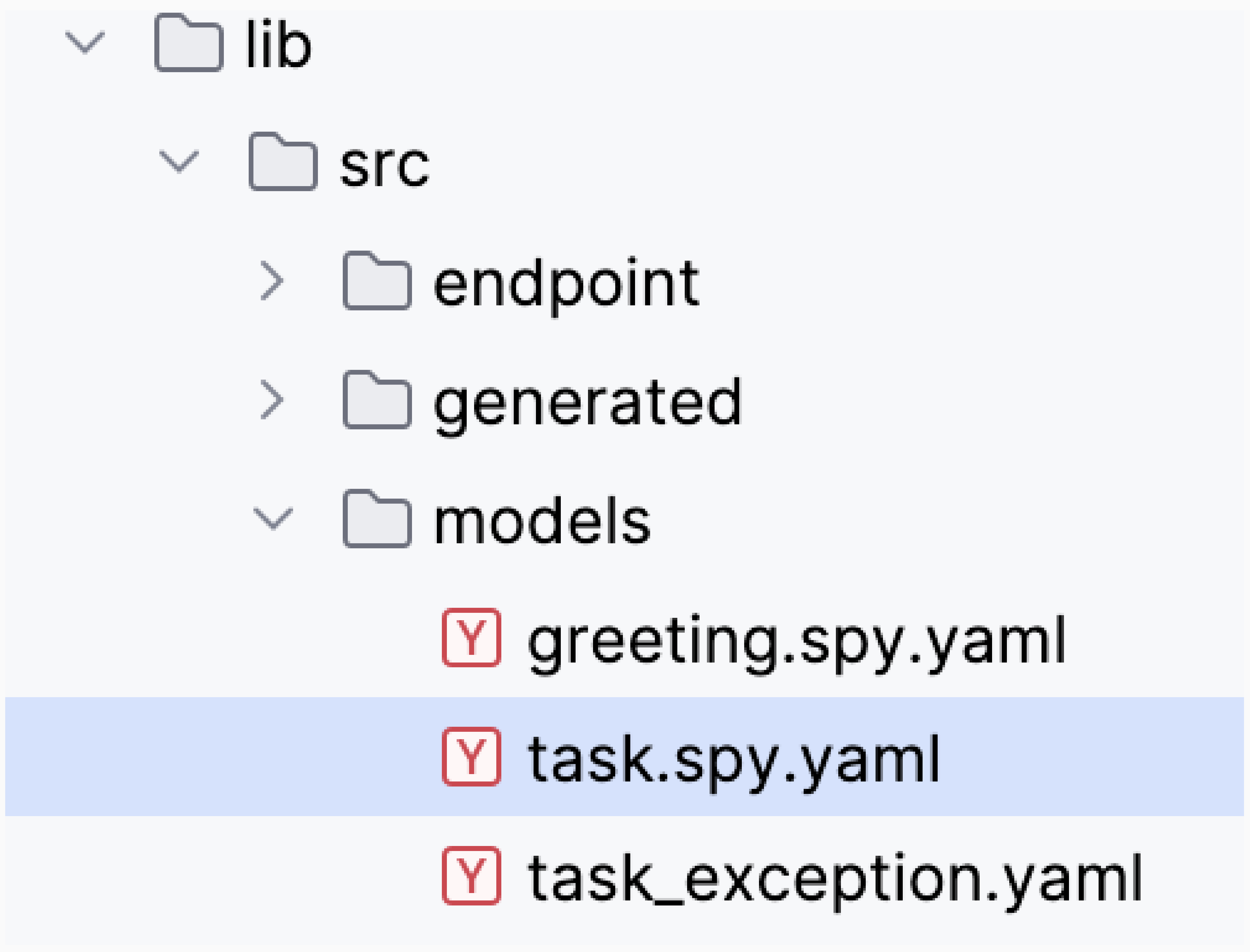


```
config
  development.yaml
  generator.yaml
  passwords.yaml
  production.yaml
  staging.yaml
deploy
  aws
    scripts
    terraform
  gcp
    console_gcr
    terraform_gce
```

Models, Endpoints & Client Calls

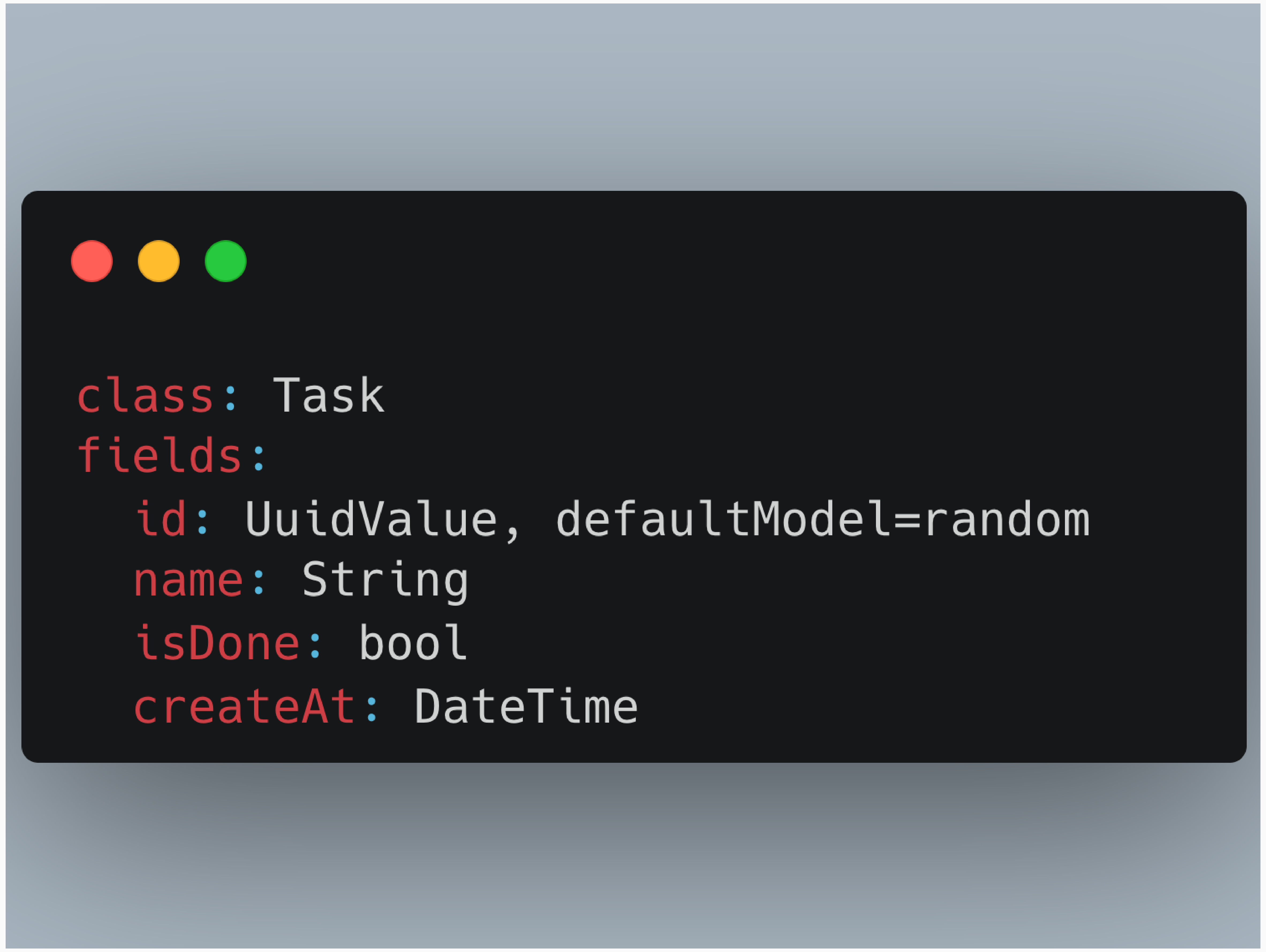
Create Model

Create file task.spy.yaml


```
lib
  src
    endpoint
    generated
    models
      greeting.spy.yaml
      task.spy.yaml
      task_exception.yaml
```

Create Model

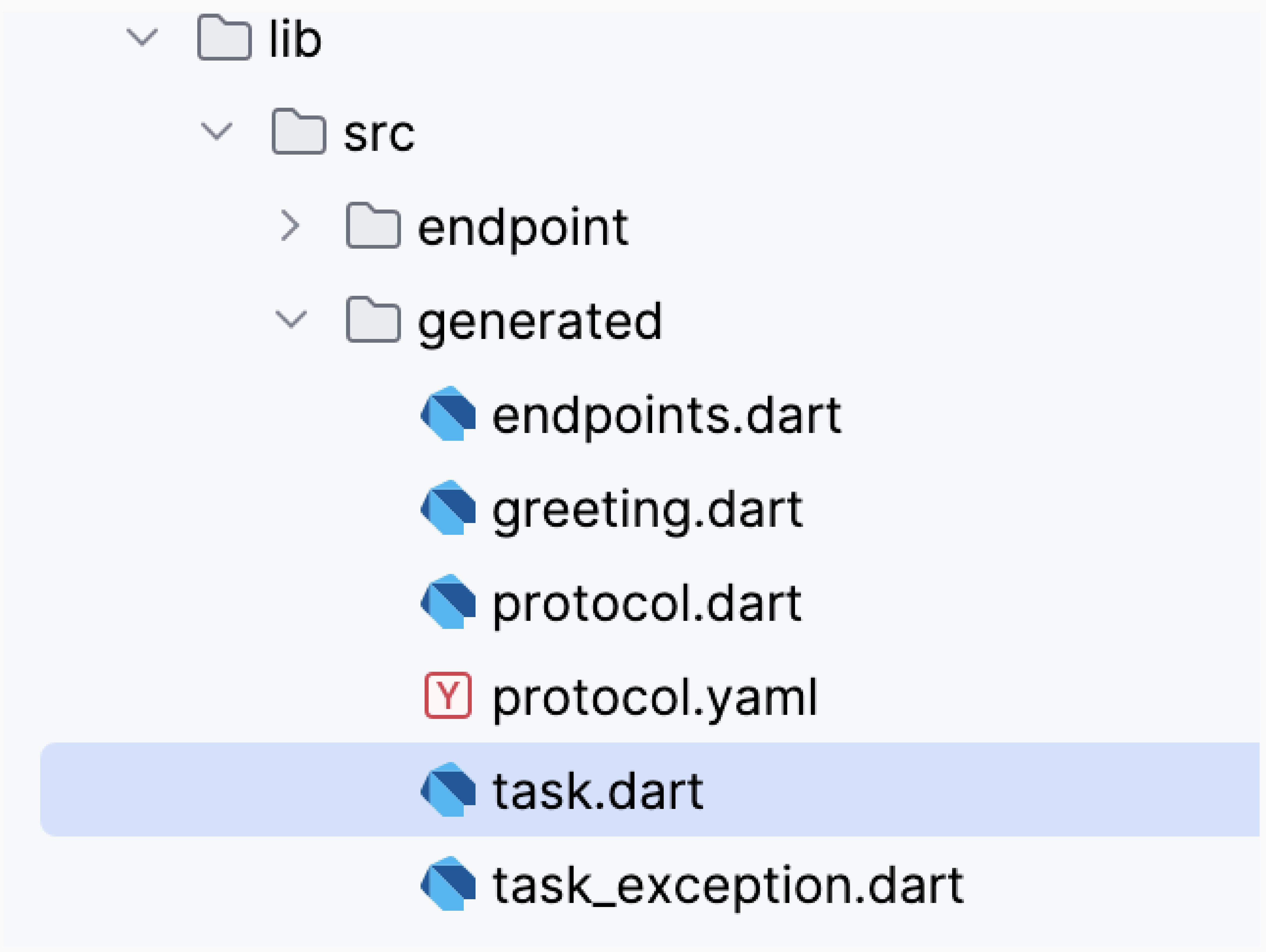
Define the model



```
class: Task
fields:
  id: UuidValue, defaultModel=random
  name: String
  isDone: bool
  createAt: DateTime
```

Create Model

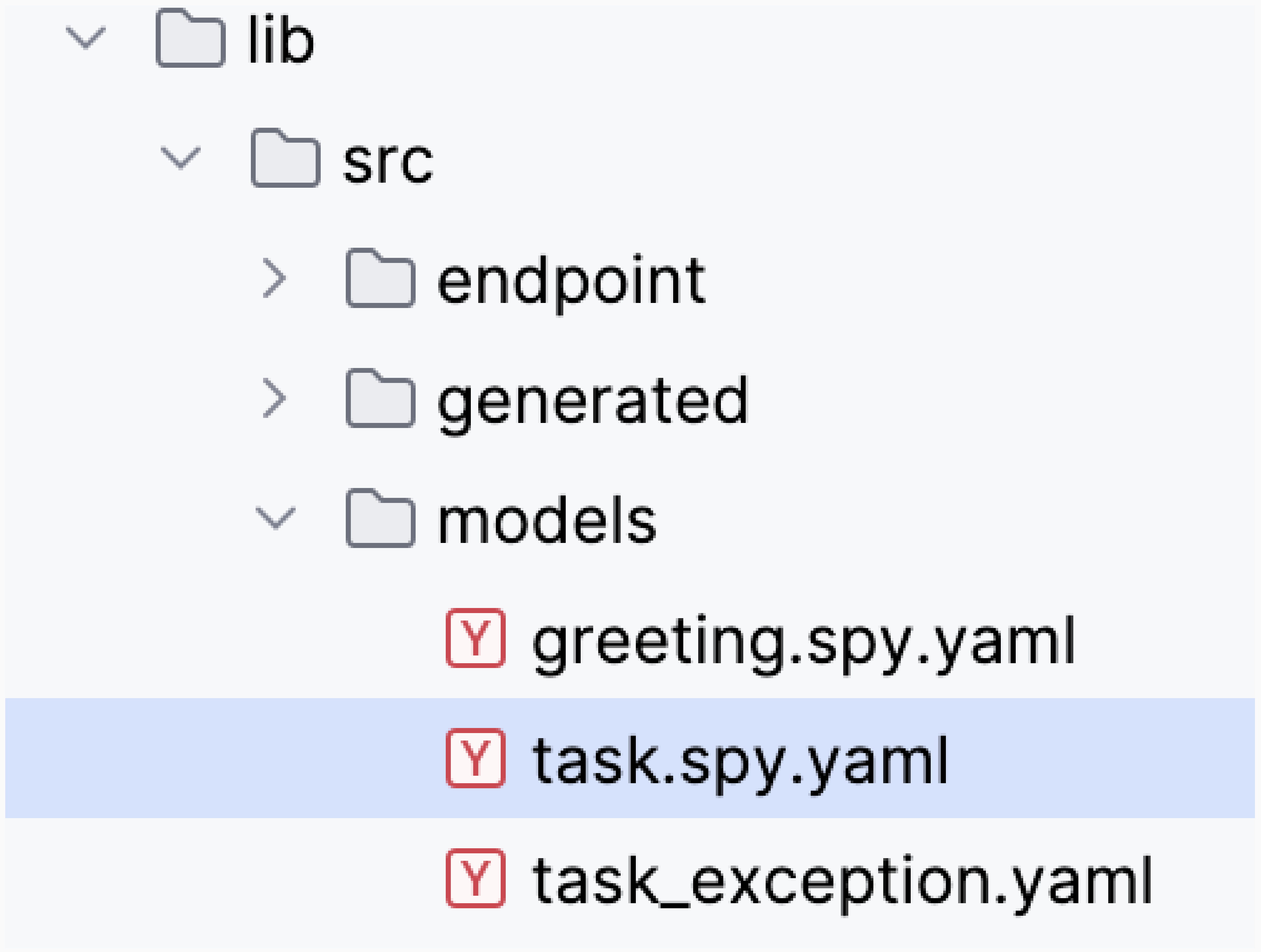
Generate the model: *serverpod generate*



```
lib
  src
    endpoint
    generated
      endpoints.dart
      greeting.dart
      protocol.dart
      protocol.yaml
      task.dart
      task_exception.dart
```

Create Table

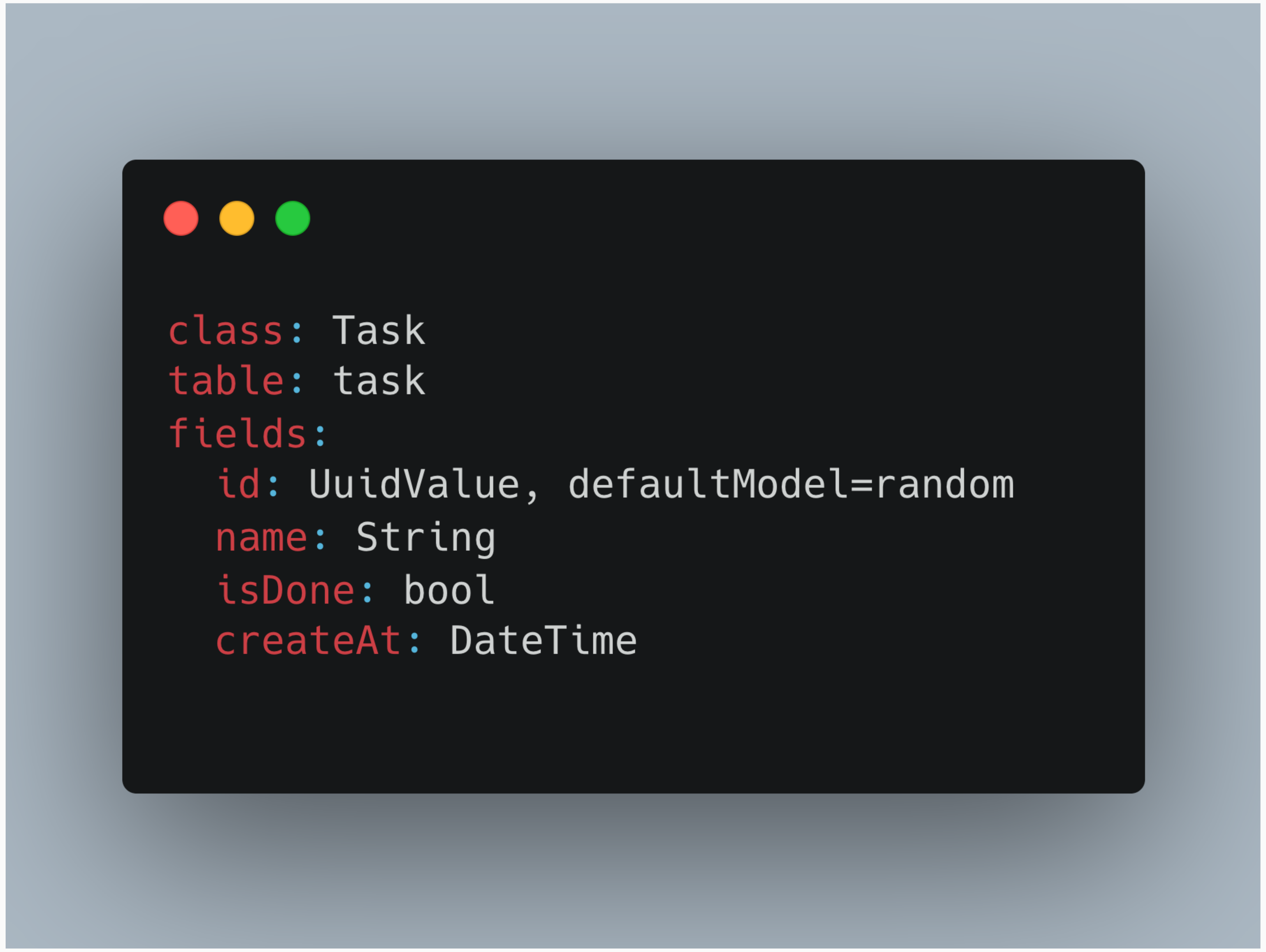
Create file task.spy.yaml (or use the same used for model)



```
lib
  src
    endpoint
    generated
    models
      greeting.spy.yaml
      task.spy.yaml
      task_exception.yaml
```

Create Table

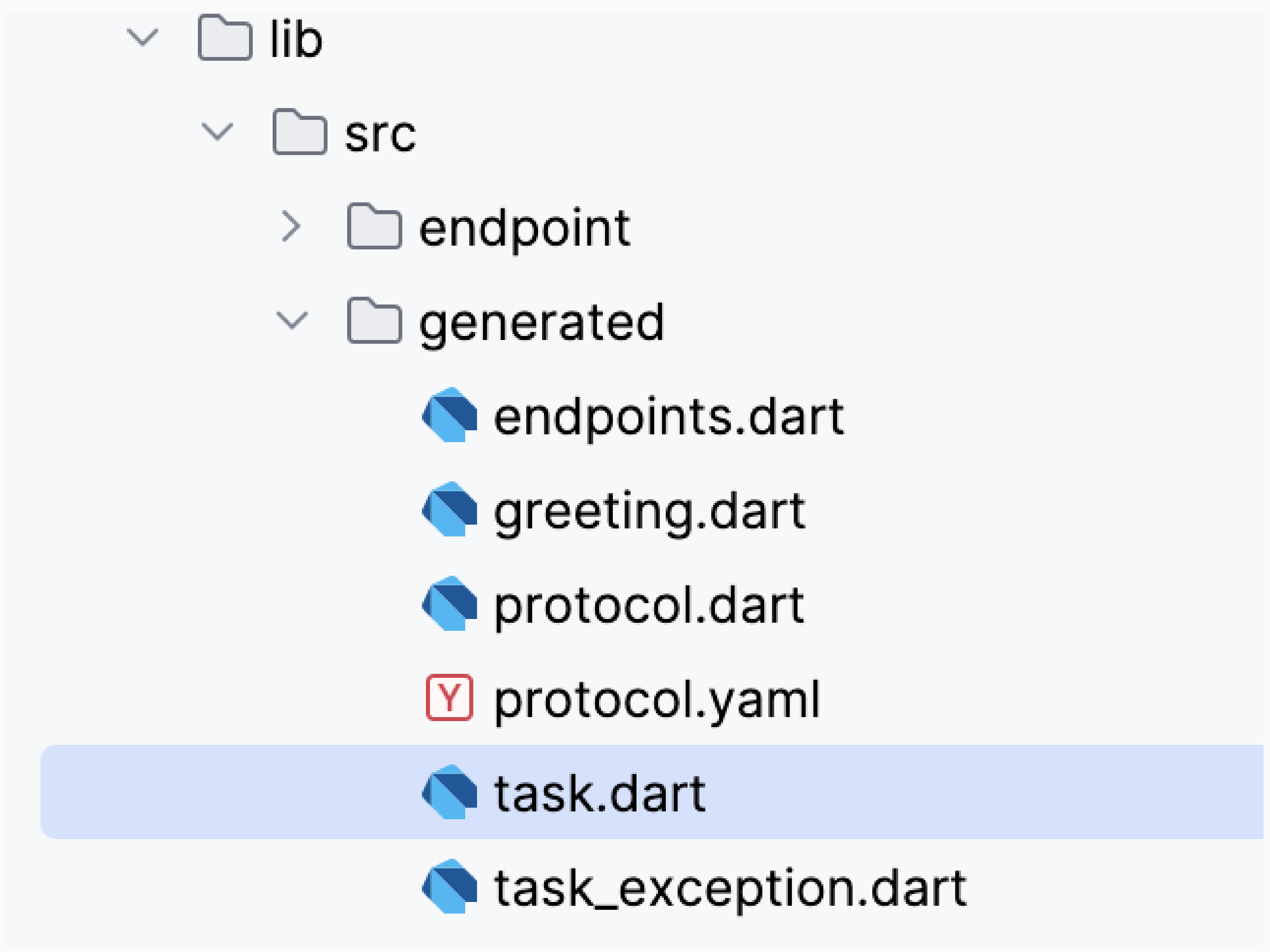
Define the **table** attribute



```
class: Task
table: task
fields:
  id: UuidValue, defaultModel=random
  name: String
  isDone: bool
  createAt: DateTime
```

Create Table

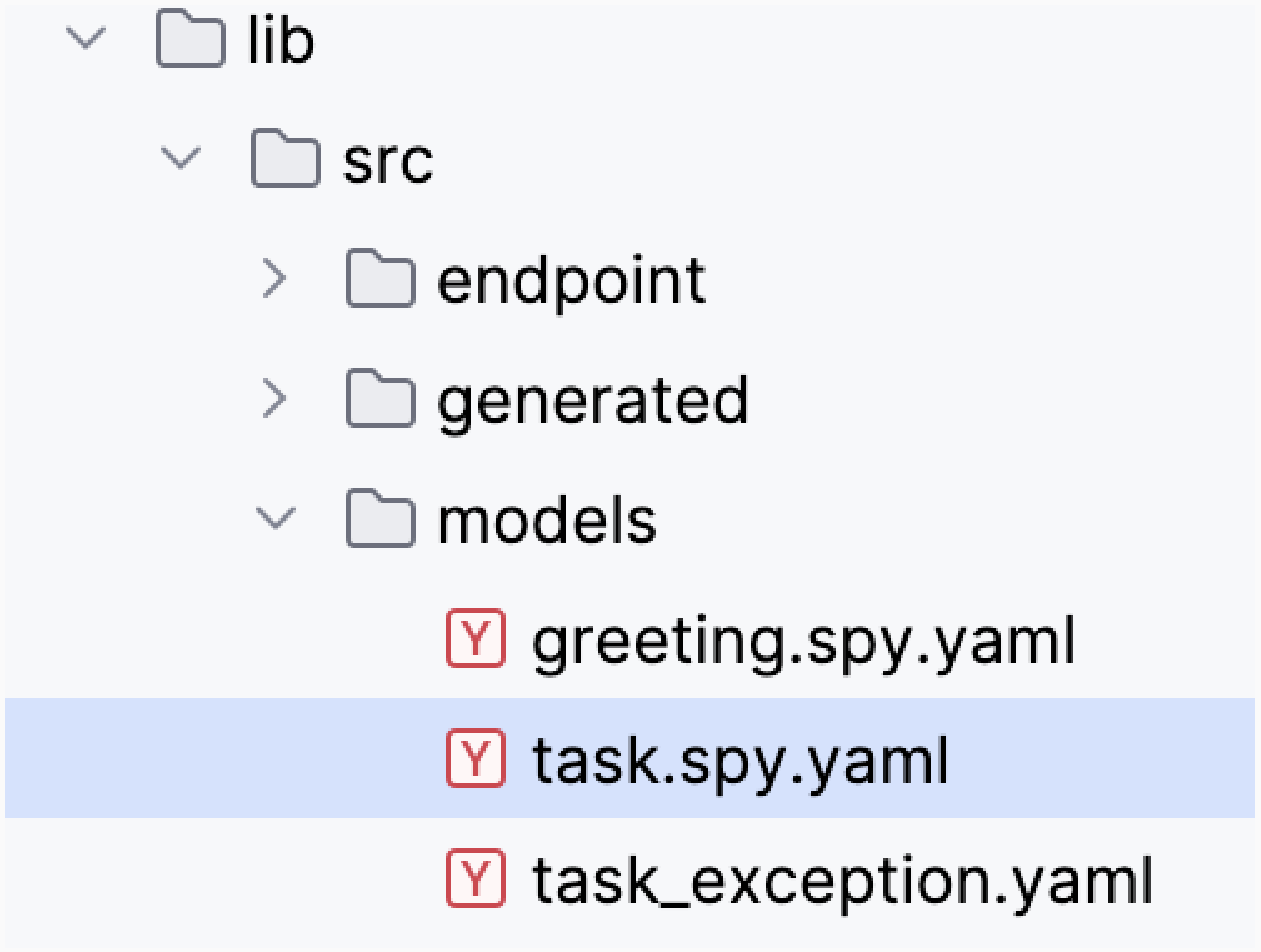
Generate the table class: *serverpod generate*
Create migration: *serverpod create-migrations*



```
lib
  src
    endpoint
    generated
      endpoints.dart
      greeting.dart
      protocol.dart
      protocol.yaml
      task.dart
      task_exception.dart
```

Create Exception

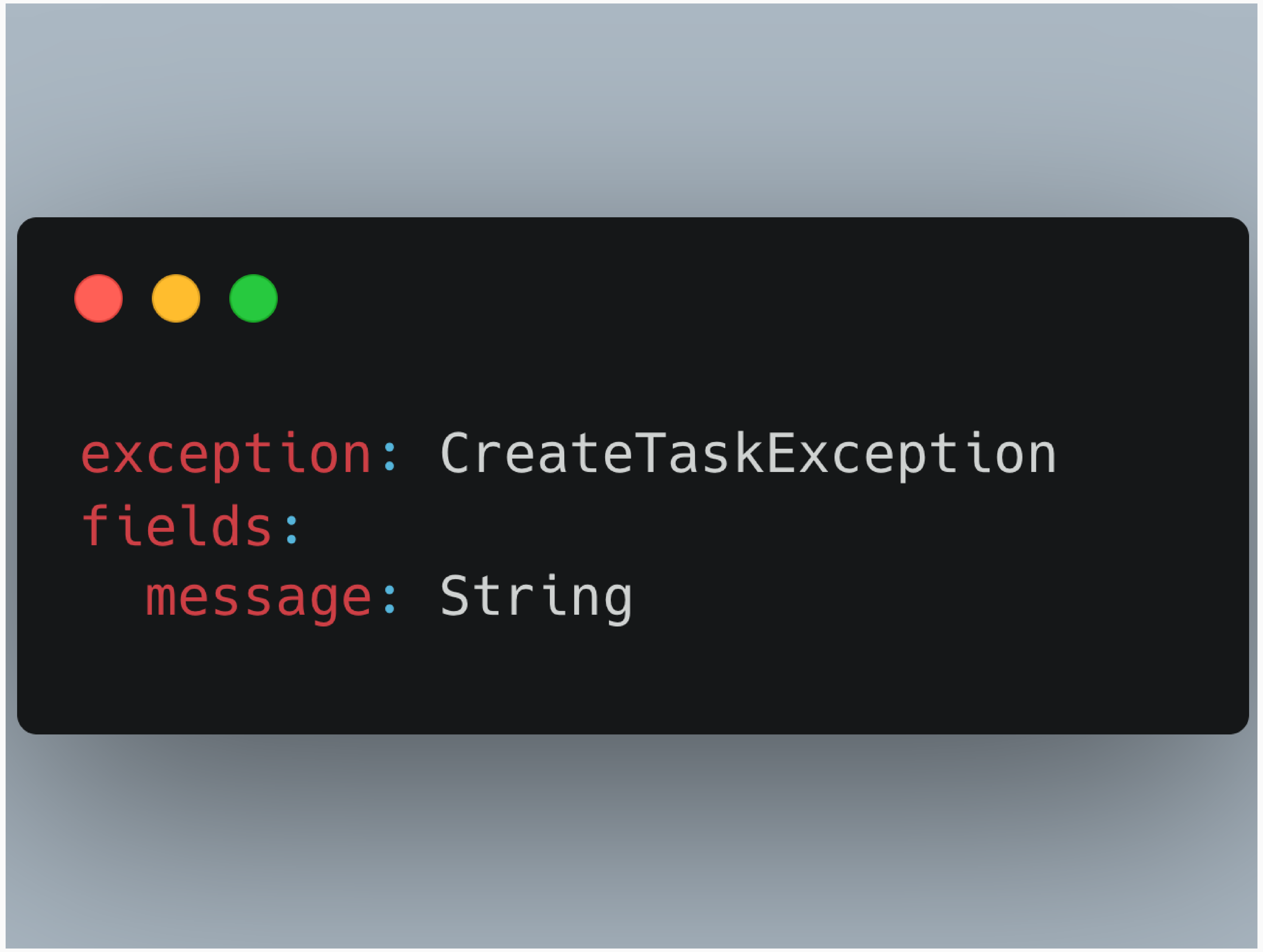
Create file task_exception.yaml



```
lib
  src
    endpoint
    generated
    models
      greeting.spy.yaml
      task.spy.yaml
      task_exception.yaml
```

Create Exception

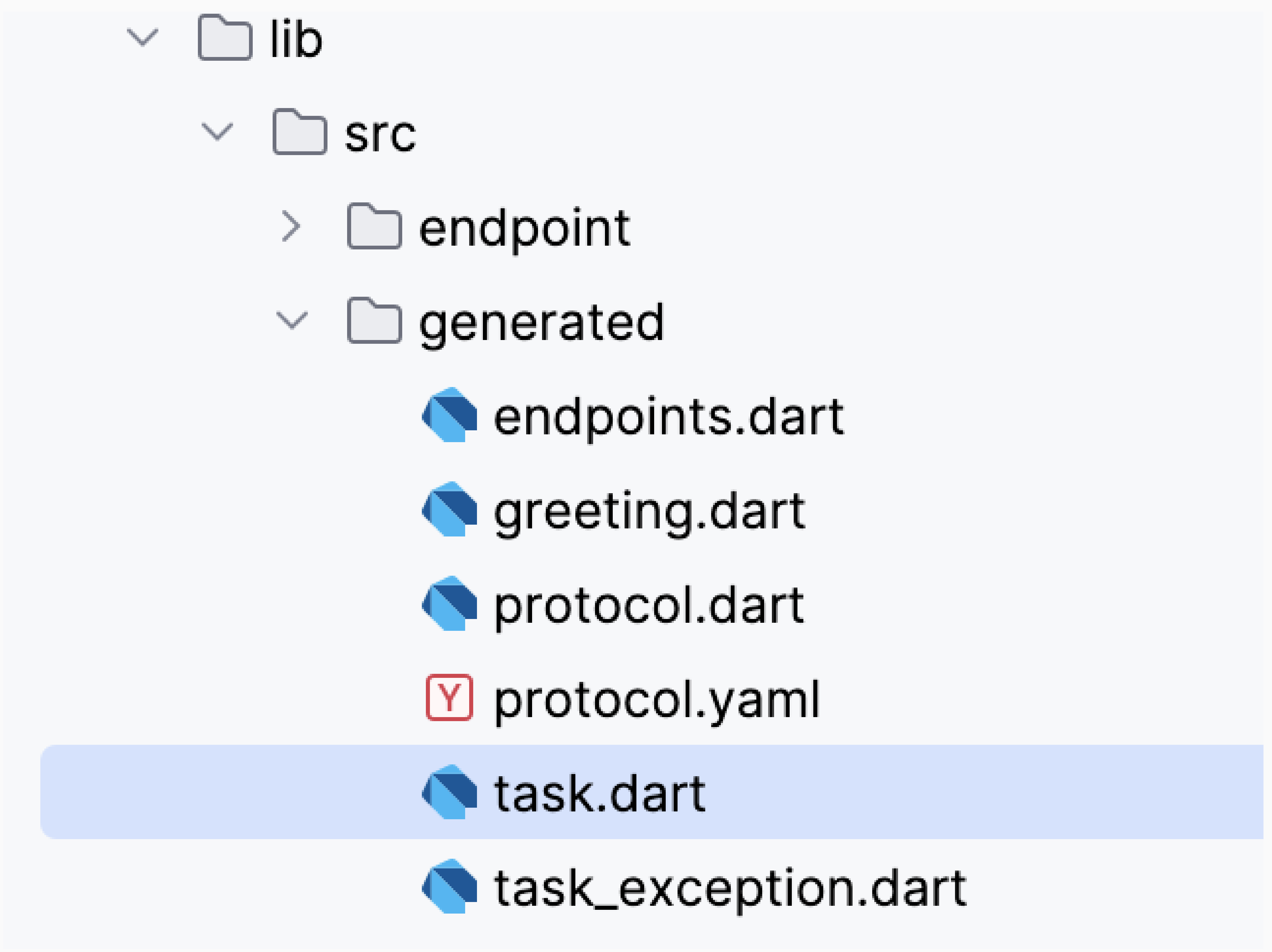
Define the **exception** attribute



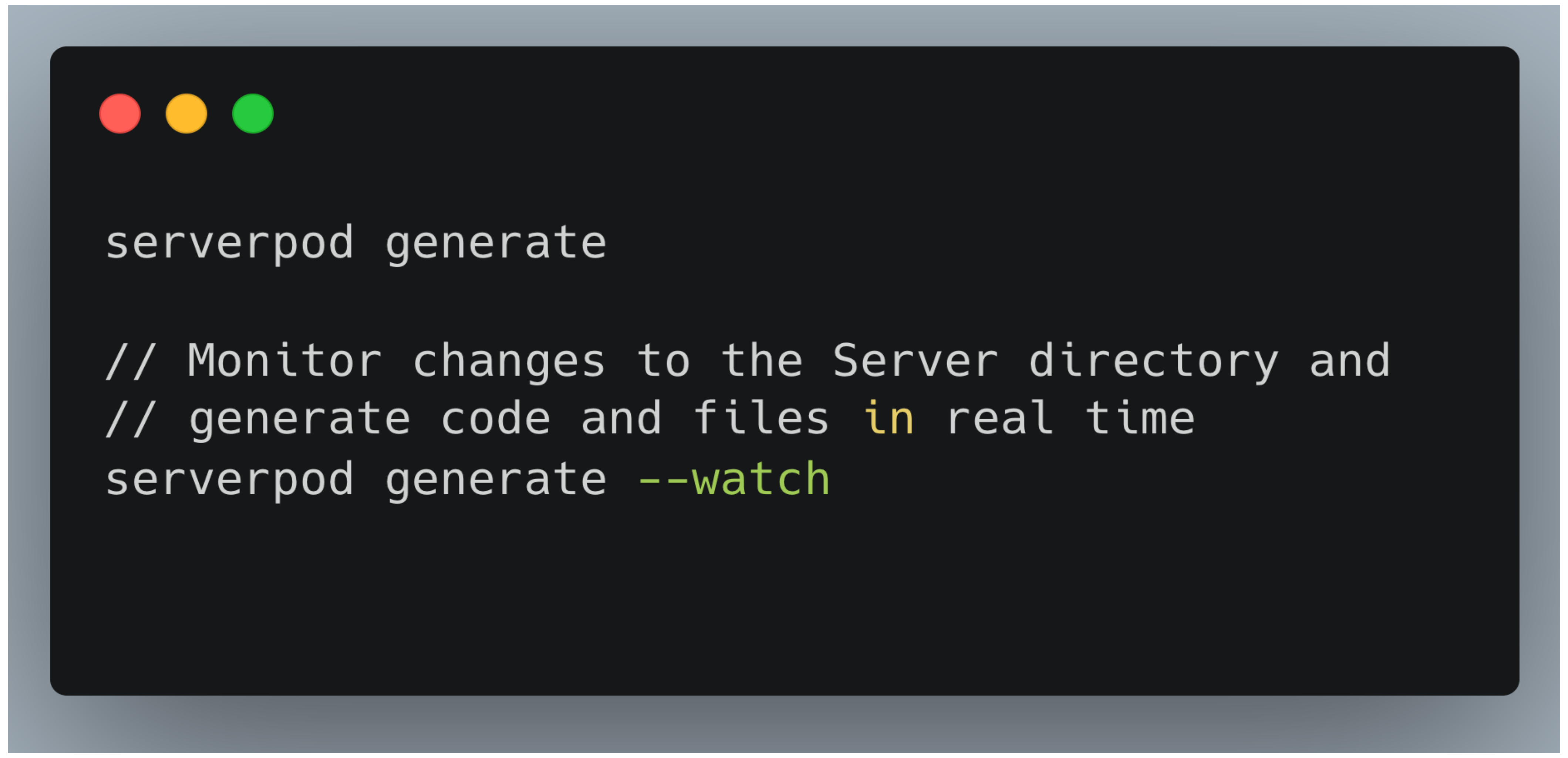
```
exception: CreateTaskException
fields:
  message: String
```

Create Exception

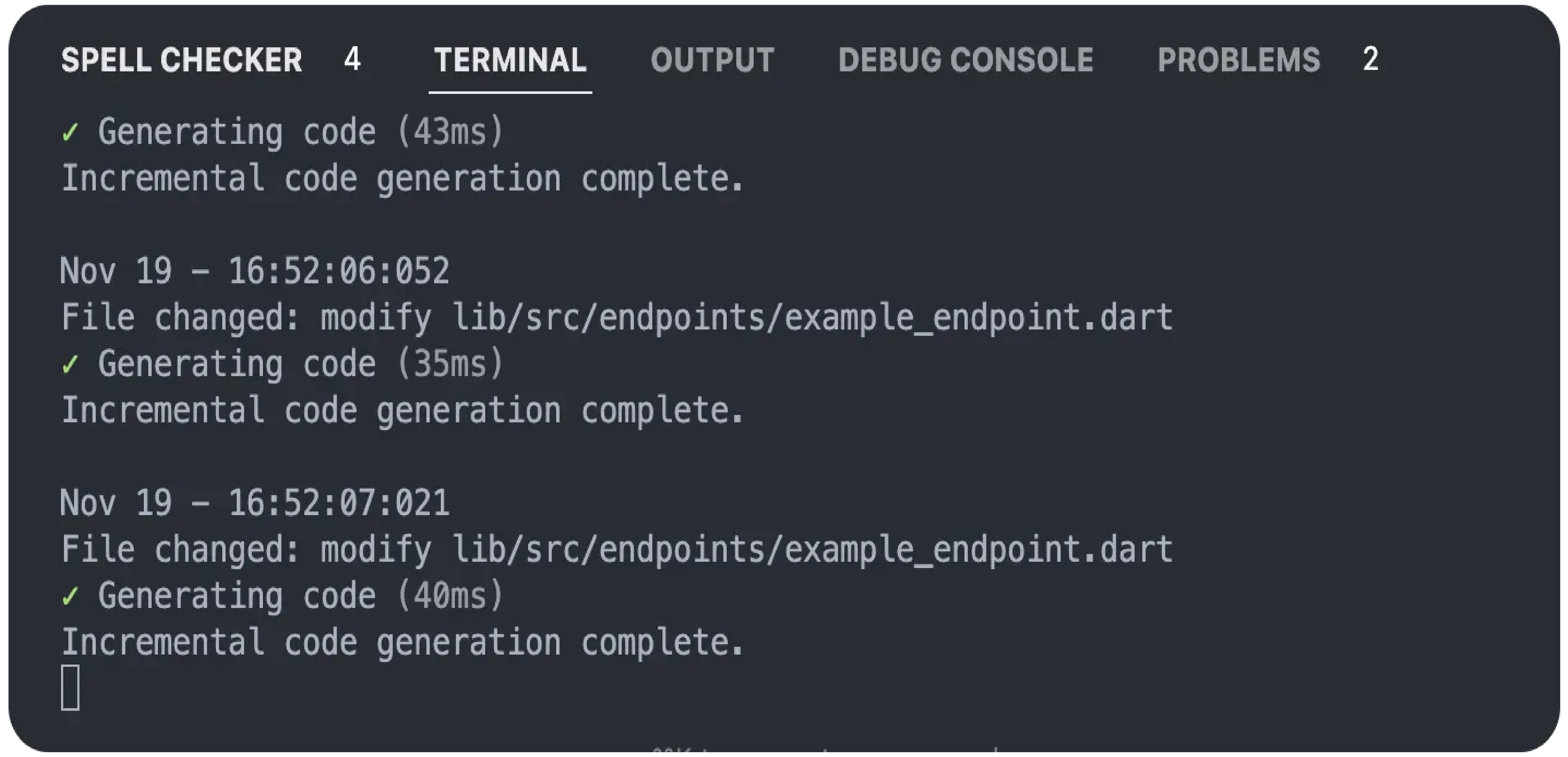
Generate the exception: *serverpod generate*


```
lib
  src
    endpoint
    generated
      endpoints.dart
      greeting.dart
      protocol.dart
      protocol.yaml
      task.dart
      task_exception.dart
```

Code generation



```
serverpod generate

// Monitor changes to the Server directory and
// generate code and files in real time
serverpod generate --watch
```

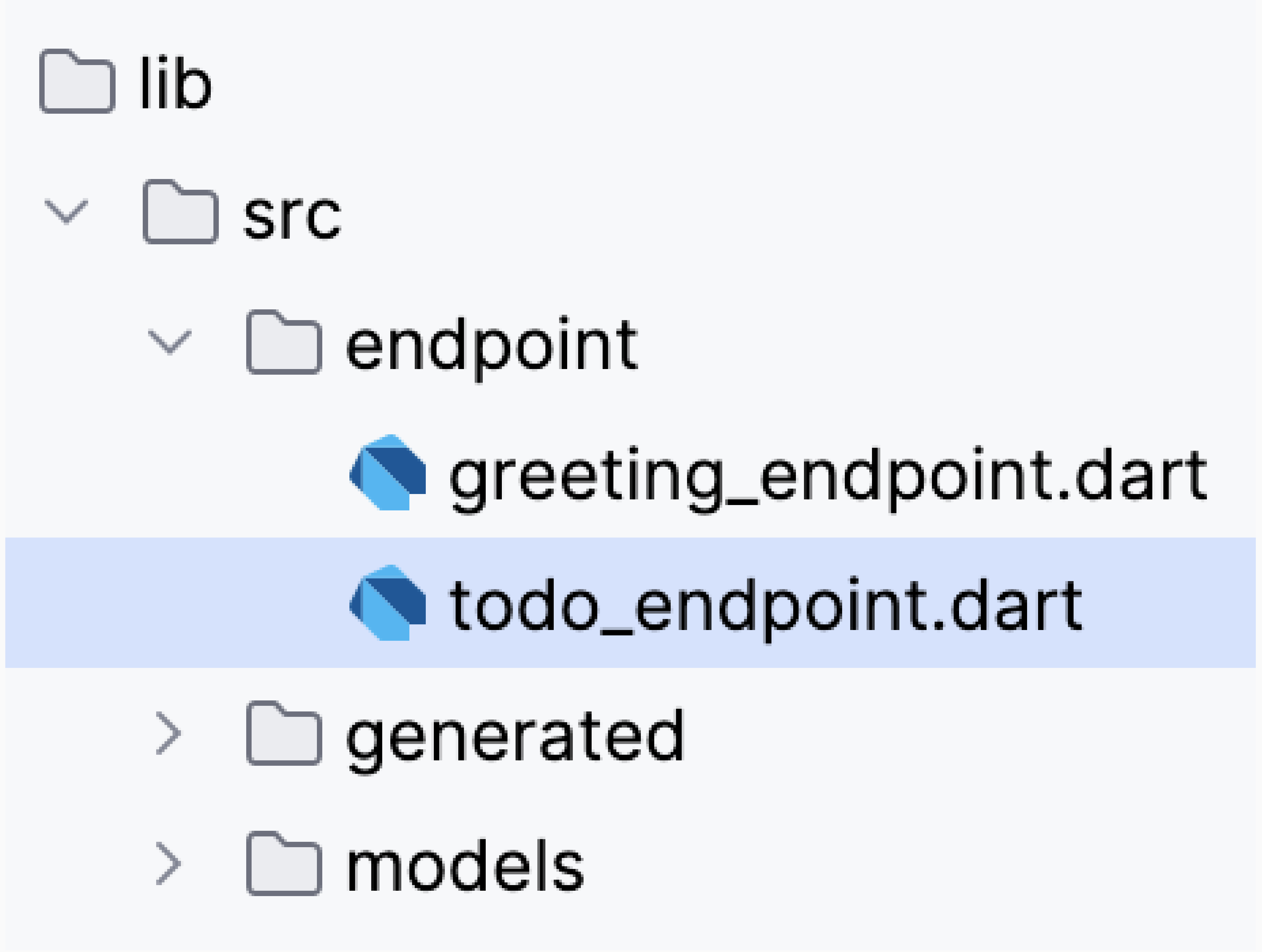


```
✓ Generating code (43ms)
Incremental code generation complete.

Nov 19 - 16:52:06:052
File changed: modify lib/src/endpoints/example_endpoint.dart
✓ Generating code (35ms)
Incremental code generation complete.

Nov 19 - 16:52:07:021
File changed: modify lib/src/endpoints/example_endpoint.dart
✓ Generating code (40ms)
Incremental code generation complete.
```

Ctrl/Cmd + S → Serverpod supports Hot Reload

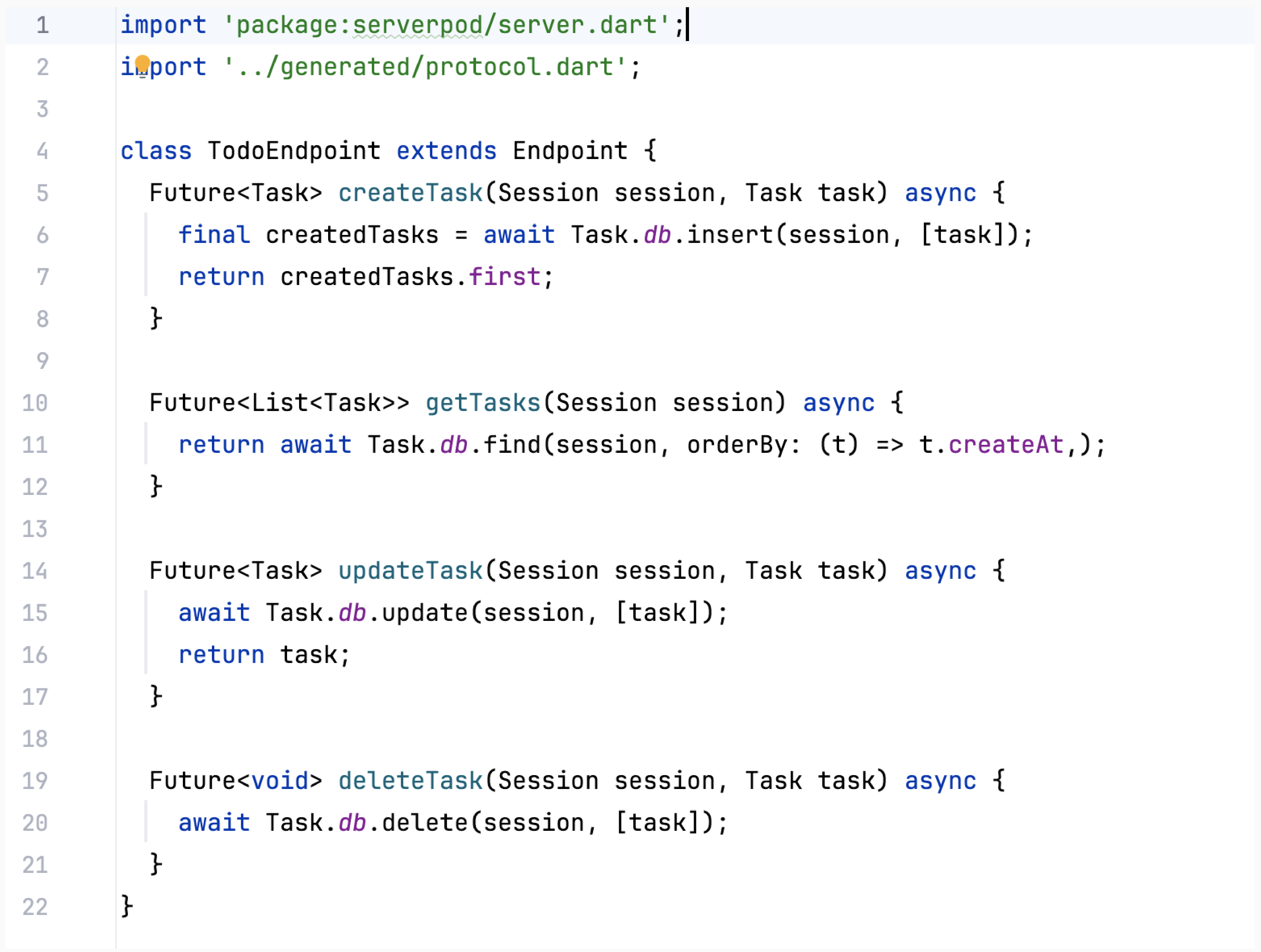
Create Endpoint

Create file todo_endpoint.dart



Create Endpoint

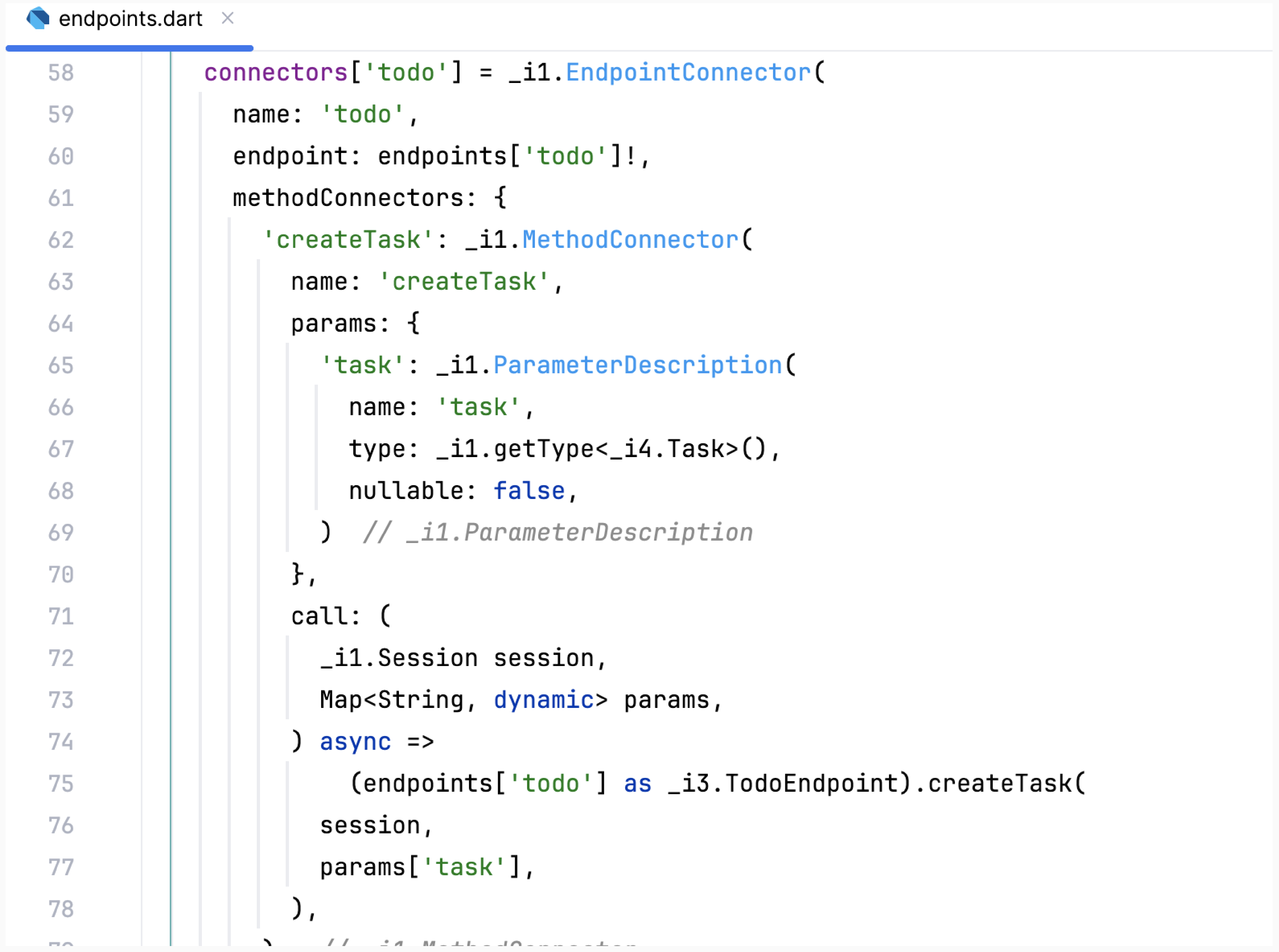
Write the endpoint class (**must extends Endpoint**)



```dart
import 'package:serverpod/server.dart';
import '../generated/protocol.dart';

class TodoEndpoint extends Endpoint {
  Future<Task> createTask(Session session, Task task) async {
    final createdTasks = await Task.db.insert(session, [task]);
    return createdTasks.first;
  }

  Future<List<Task>> getTasks(Session session) async {
    return await Task.db.find(session, orderBy: (t) => t.createAt,);
  }

  Future<Task> updateTask(Session session, Task task) async {
    await Task.db.update(session, [task]);
    return task;
  }

  Future<void> deleteTask(Session session, Task task) async {
    await Task.db.delete(session, [task]);
  }
}
```

Create Endpoint

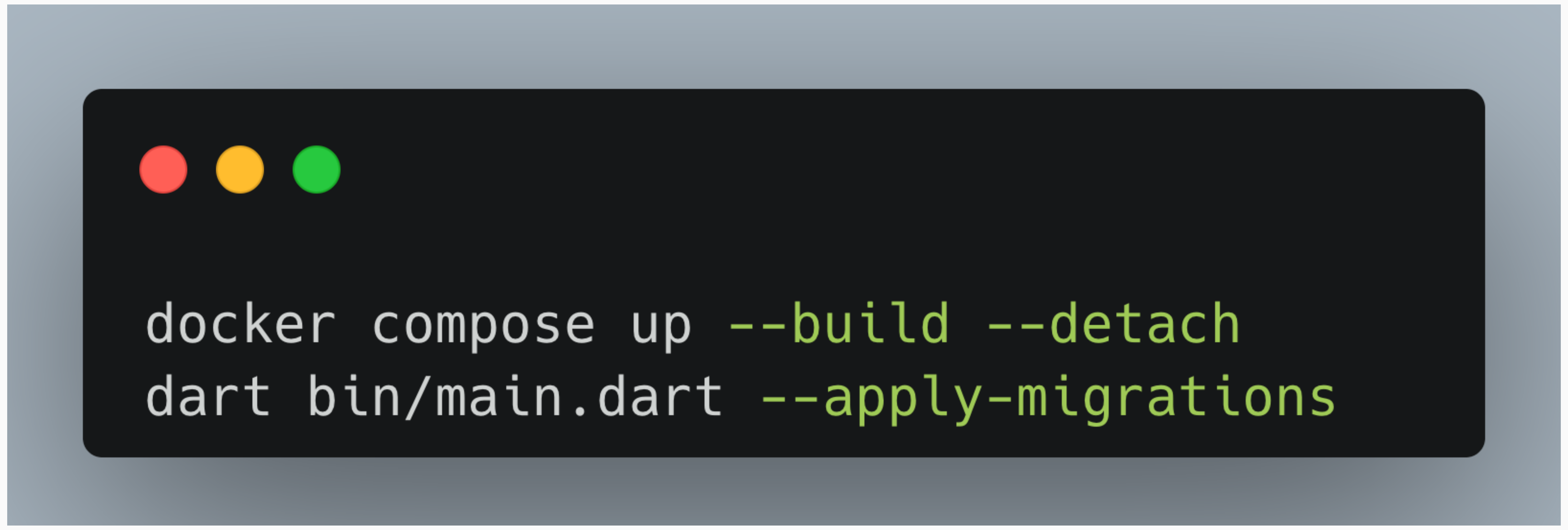
Generate Server and Client code:*serverpod generate*



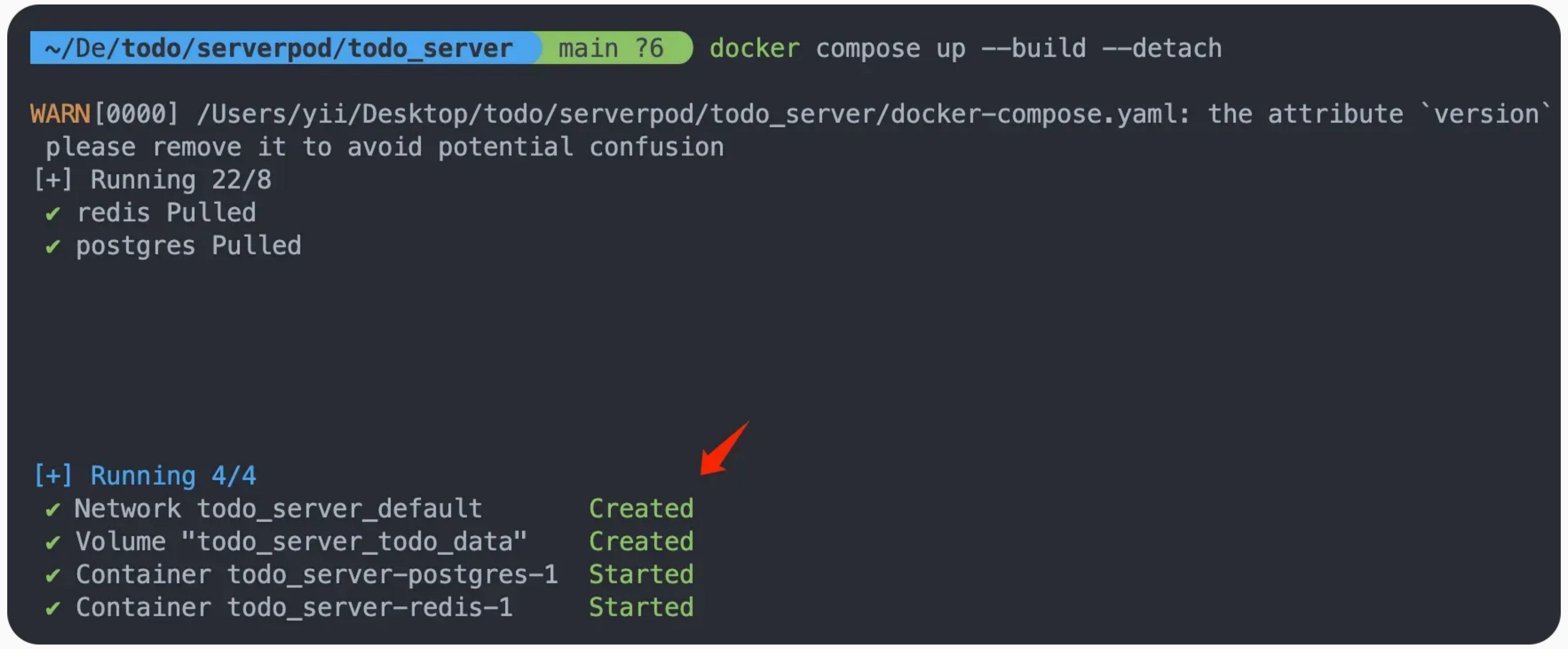
endpoints.dart

```dart
connectors['todo'] = _i1.EndpointConnector(
  name: 'todo',
  endpoint: endpoints['todo']!,
  methodConnectors: {
    'createTask': _i1.MethodConnector(
      name: 'createTask',
      params: {
        'task': _i1.ParameterDescription(
          name: 'task',
          type: _i1.getType<_i4.Task>(),
          nullable: false,
        )  // _i1.ParameterDescription
      },
      call: (
        _i1.Session session,
        Map<String, dynamic> params,
      ) async =>
          (endpoints['todo'] as _i3.TodoEndpoint).createTask(
        session,
        params['task'],
      ),
    ), // _i1.MethodConnector
```

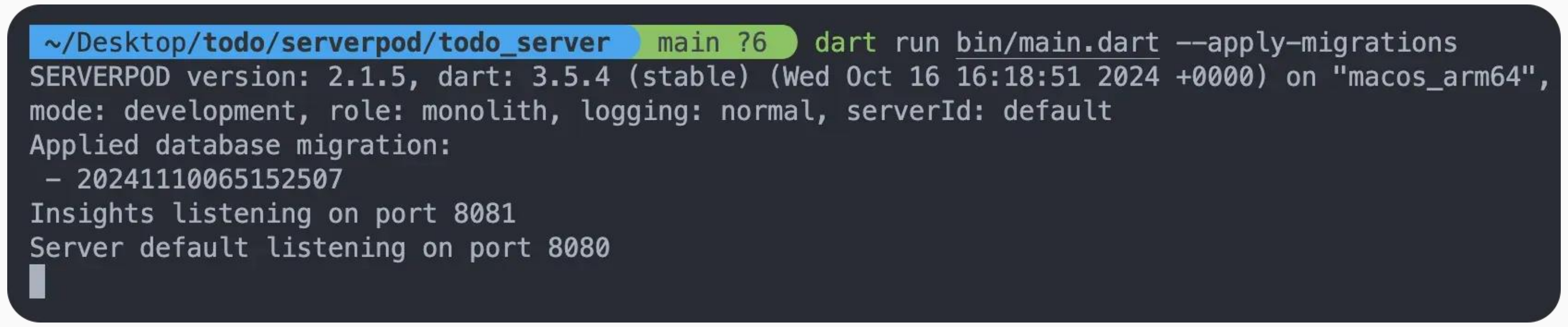
Server running



```
docker compose up --build --detach
dart bin/main.dart --apply-migrations
```



```
~/De/todo/serverpod/todo_server  main ?6  docker compose up --build --detach

WARN[0000] /Users/yii/Desktop/todo/serverpod/todo_server/docker-compose.yaml: the attribute `version`
 please remove it to avoid potential confusion
[+] Running 22/8
 ✔ redis Pulled
 ✔ postgres Pulled




[+] Running 4/4
 ✔ Network todo_server_default       Created
 ✔ Volume "todo_server_todo_data"    Created
 ✔ Container todo_server-postgres-1  Started
 ✔ Container todo_server-redis-1     Started
```



```
~/Desktop/todo/serverpod/todo_server  main ?6  dart run bin/main.dart --apply-migrations
SERVERPOD version: 2.1.5, dart: 3.5.4 (stable) (Wed Oct 16 16:18:51 2024 +0000) on "macos_arm64",
mode: development, role: monolith, logging: normal, serverId: default
Applied database migration:
 - 20241110065152507
Insights listening on port 8081
Server default listening on port 8080
```

Client

```
final client = Client('<http://$localhost:8080/>')
  ..connectivityMonitor = FlutterConnectivityMonitor();
```

● ○ ○ ○ ○ ○

Client

```
void getTasks() async {
  final result = await client.todo.getTasks();

  setState(() {
    tasks.addAll(result);
  });
}
```

Client

```
void createTask() async {
  final taskName = _textEditingController.text;
  final task = Task(
    name: taskName,
    isDone: false,
    createAt: DateTime.now(),
  );  // Task

  try {
    final createdTask = await client.todo.createTask(task);
```

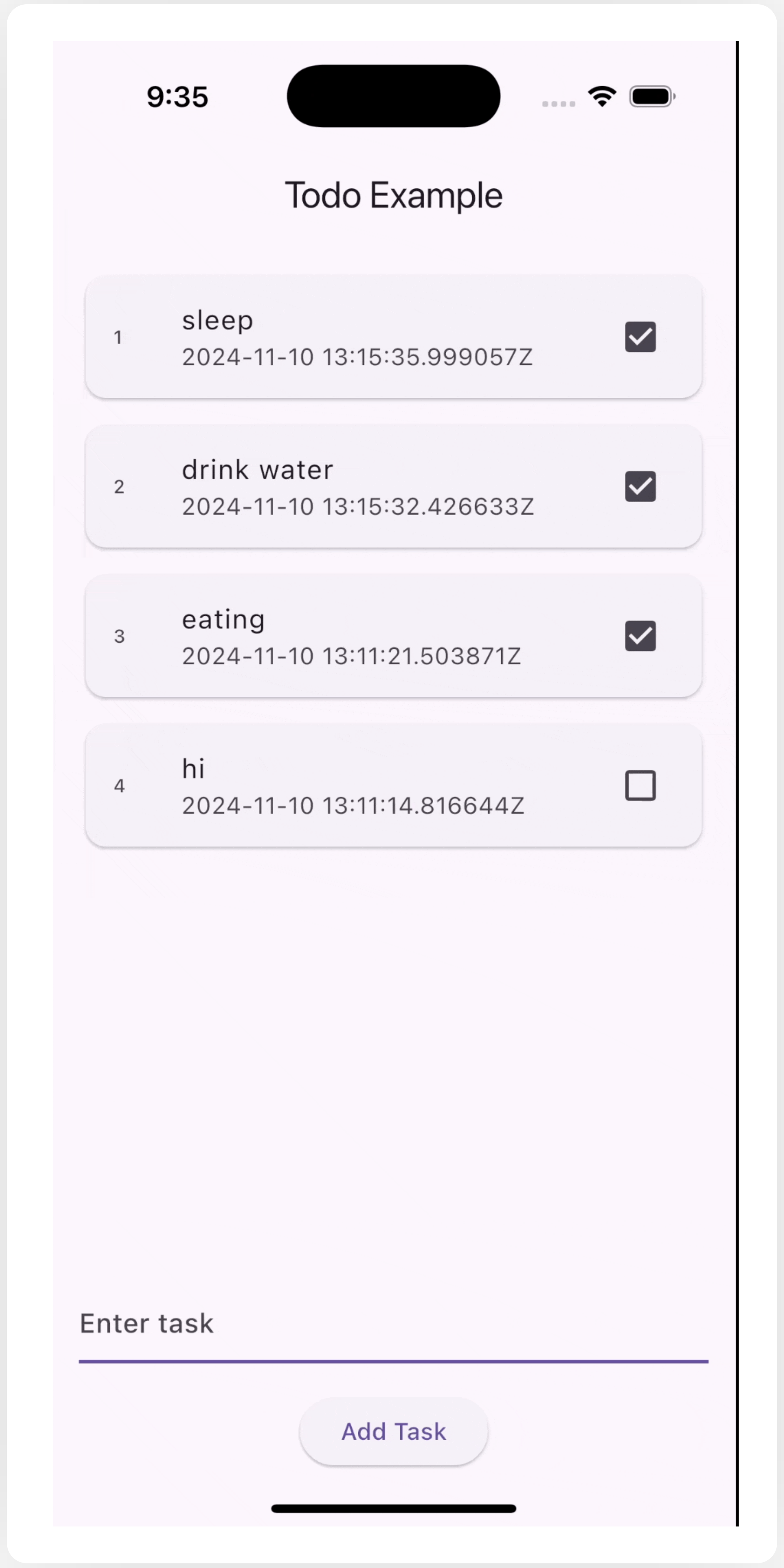
Client

```
Future<void> updateTask(int index) async {
  final task = tasks.elementAt(index);
  final newTask = task.copyWith(isDone: !task.isDone);

  try {
    setState(() {
      tasks[index] = newTask;
    });

    await client.todo.updateTask(newTask);
```

Client

```
void deleteTask(int index) async {
  final task = tasks.elementAt(index);

  try {
    setState(() {
      tasks.remove(task);
    });

    await client.todo.deleteTask(task);
```

Client demo


9:35

Todo Example

1	sleep 2024-11-10 13:15:35.999057Z	✓
2	drink water 2024-11-10 13:15:32.426633Z	✓
3	eating 2024-11-10 13:11:21.503871Z	✓
4	hi 2024-11-10 13:11:14.816644Z	☐

Pros, Cons & Conclusion

PROS, CONS & CONCLUSION

	Serverpod	Others (e.g., Node.js, Firebase)
Language	Dart only	JS, TS, Go, Python, etc.
Typing	Strong, end-to-end	Often weak or needs extra libraries
Flutter Integration	**Native**	Manual REST/GraphQL
Tooling	**Built-in CLI & UI**	Varies by framework
Learning Curve	**Low for Flutter devs**	May need full-stack knowledge
Scalability & Flexibility	Good, less modular	Highly flexible
Adoption	**Still limited**	Widely used in production

When adopt it?

DO

- Team Flutter-centric
- Fast and consistent MVPs
- Avoid context switching

DON' T

- Very complex or legacy backend
- Highly scalable microservices

Where to go now

1 Caching

2 Logging

3 Modules

4 Authentication

5 Uploading files

6 Scheduling

7 Streams

8 Serverpod mini

Conclusions

Dart full-stack is a reality with Serverpod

Simplified client-server communication

Consistent experience with powerful tooling

ME AS A
DART DEVELOPER
FLUTTER
FRONTEND
SERVERPOD
FULL-STACK POWER

Federico Parezzan

Keep in touch:

federico.parezzan@outlook.it

You can find me:

Portfolio Linkedin Github