



Document in readme

Mathias Fischler authored 15 hours ago

Unverified

 **README.md** 2.02 KiB

DSys Challenge Task

by Mathias F. und Simon D.

Quickstart (Development)

```
dotnet run --project ChallengeTask.AppHost
```

Note the underlined url in the shell - it is clickable and will lead you to the dashboard:

```
→ dsys-challenge-task git:(main) X dotnet run --project ChallengeTask.AppHost
Using launch settings from ChallengeTask.AppHost\Properties\launchSettings.json...
Building...
info: Aspire.Hosting.DistributedApplication[0]
  Aspire version: 9.2.1+b590865a294feaff82f06c4fedef62ba1fad2271
info: Aspire.Hosting.DistributedApplication[0]
  Distributed application starting.
info: Aspire.Hosting.DistributedApplication[0]
  Application host directory is: C:\Users\fischler\Documents\repos\dsys-challenge-task\ChallengeTask.AppHost
info: Aspire.Hosting.DistributedApplication[0]
  Now listening on: https://localhost:17171
info: Aspire.Hosting.DistributedApplication[0]
  Login to the dashboard at https://localhost:17171/login?t=d8f102e807a6a3bd189a57c533f0d5aa
info: Aspire.Hosting.DistributedApplication[0]
  Distributed application started. Press Ctrl+C to shut down
```

https://localhost:17171/login?
t=d8f102e807a6a3bd189a57c533f0d5aa
Strg+Klicken, um dem Link zu folgen

On the dashboard you're sent to, you can watch the state, check the logs, or even stop and restart containers/parts.

Deployment

- Use something like <https://github.com/prom3theu5/aspirational-manifests> for automation (untested)
- Use dotnet publish (untested)

Architecture Remarks

- The ChallengeTask.Api project contains a load balancer and reverse proxy (YARP)
- The ChallengeTask.AppHost contains the .NET-Aspire configuration and setup, it is the entry point
- The challenge-task-ui folder contains the svelte-driven ui project, hosted by vite in development
- The PythonApi folder contains the API communicating with the ui, to manipulate files and subscribe to updates

SSE

The python project uses FastAPI to serve the routes:

- It uses a FastAPIs StreamingResponse with a text/event-stream based format to provide an SSE endpoint
 - See return StreamingResponse(generator, media_type="text/event-stream")
- It uses peewee as ORM and SQLite as database (SQLite is not production ready!)
 - The SQLite database file is linked into the docker container and shared
- To track changes and notify connected users watchfiles is used to watch the SQLite file
 - A proper database would have a notification feature with a callback, but SQLite's API for that does not work when updates are pushed from another process

Load balancing

- YARP is configured to use Round-Robin load balancing
- It also uses active Health-Checks - meaning an endpoint on each python API server is called regularly
- If a server is unhealthy (unreachable or takes too long) it is removed from the cluster and not used as a destination until it is healthy again
- The ui restarts the SSE connection to the server if the connection is closed
 - Reconciliation is communicated to the user, no interaction is needed