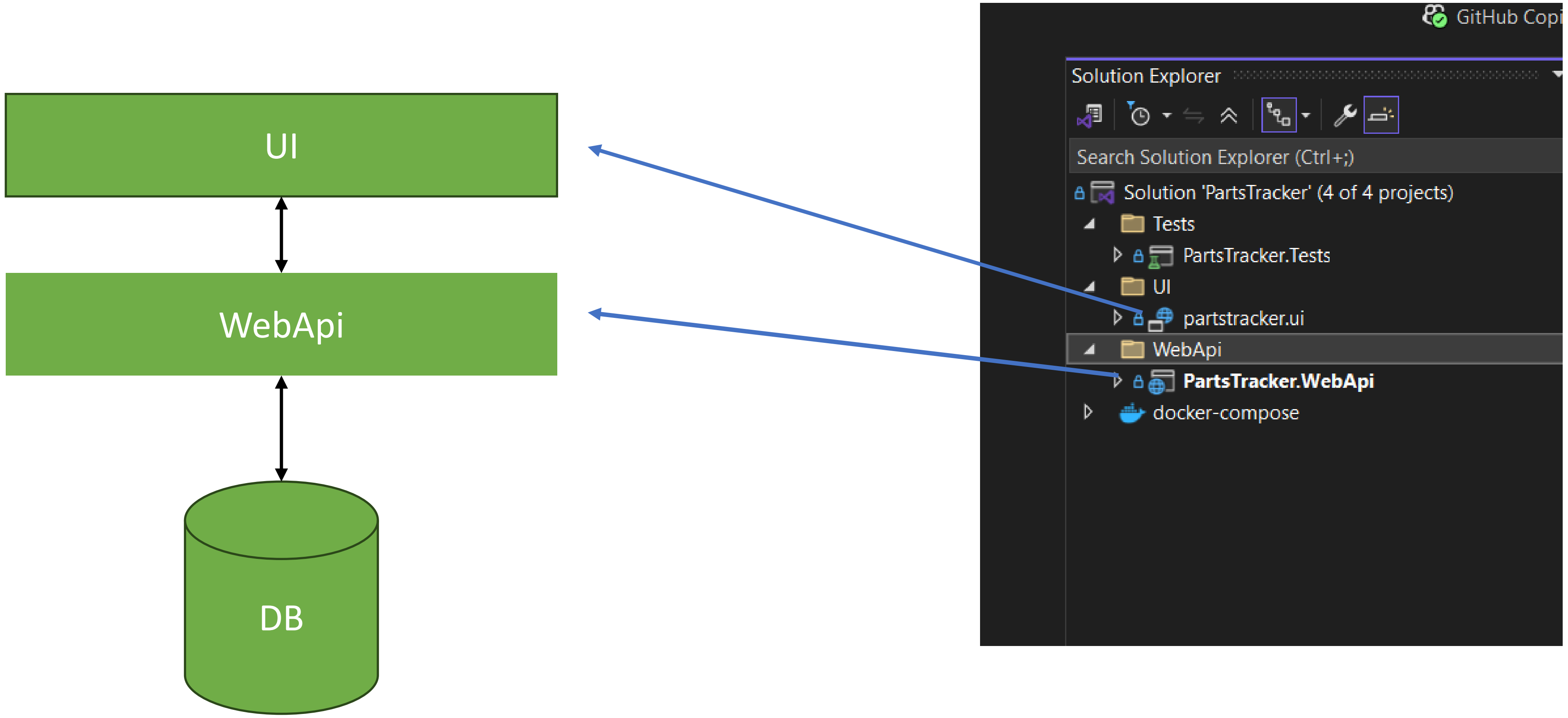
PartsTracker

Rudi Grobler

High Level Architecture

My solution is a very simple "microservices-inspired" architecture, with a .NET Web API backend, a separate UI, and infrastructure as code (IaC) managed via Terraform



Cloud-native

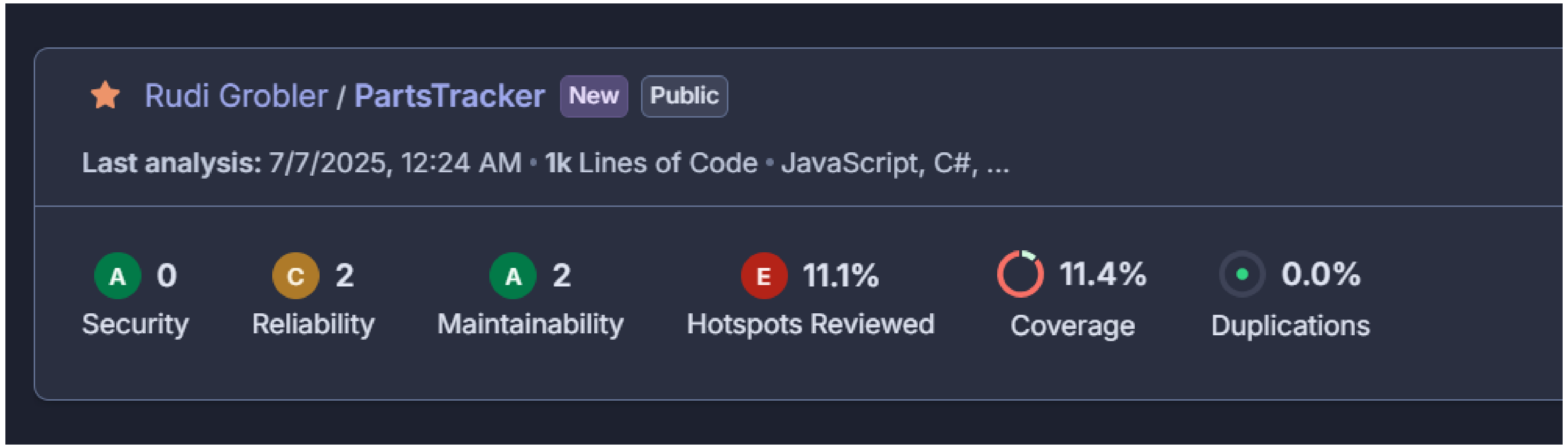
- The API is stateless, making it easy to scale horizontally in the cloud.
- The use of Docker for containerization, enabling deployment to any cloud provider.
- Infrastructure is defined as code, supporting repeatable, automated deployments.
- API, UI, and infrastructure are decoupled, allowing independent development and scaling.

Constraints

- To make this as easy as possible, currently theirs no authentication and all communication is via HTTP.
- Please note this is only for development, and should NEVER be deployed like this!

Code Quality

- All code are checked via SonarCloud, triggered from a GitHub action to ensure we have no bugs, code smells and good coverage.



Rudi Grobler / PartsTracker New Public

Last analysis: 7/7/2025, 12:24 AM • 1k Lines of Code • JavaScript, C#, ...

A 0 Security

C 2 Reliability

A 2 Maintainability

E 11.1% Hotspots Reviewed

11.4% Coverage

0.0% Duplications

- Although not "production ready", the start of some CI/CD pipelines to automate testing and deployment, reducing risk of regressions.

Notes

- Health checks implemented.
- RESTful conventions used in controllers.
- DbContext (EF) and a simple repository pattern is defined and used via Dependency Injection.
- All end-points are well documented and exposed via OpenAPI.
- Logging is actively used to help debug potential issues.
- UI is a simple react app, using bootstrap
  - TODO: Client side validation