

WaterWizards

Justin Dewitz, Erick Zeiler, Max Kondratov, Julian jNachnamej

17. Juli 2025



Inhaltsverzeichnis

- Einleitung
- Organisation
- Technologie und Architektur
- DevOps
- Technische Erfahrungen
- Analyse
- Wiki
- Erfahrungen und Fazit

Einleitung

- ▶ Was ist WaterWizards?
 1. Multitplayer Real-Time Schiffe versenken
 2. Angriff durch Zauber, die durch Karten repräsentiert werden
 3. Ziel: Zerstörung der gegnerischen Schiffe
- ▶ Warum WaterWizards?
 1. Schiffe versenken ist ein Klassiker
 2. Durch Real-Time wird es dynamischer
 3. Für jede Altersgruppe interessant
- ▶ Was macht WaterWizards besonders?
 1. Kombination aus Strategie und schnellen Entscheidungen
 2. Zauber und Karten bringen neue Dynamik ins Spiel
 3. Multiplayer und Echtzeit sorgen für Spannung
- ▶ Für wen ist das Spiel gedacht?
 1. Strategie-Fans
 2. Familien und Freunde
 3. Alle Altersgruppen

Organisation

- ▶ git über GitHub für die Versionsverwaltung
- ▶ Scrum mit 2-Wochen-Sprints
- ▶ Kanban/Issue-Board über GitHub
- ▶ Kommunikation über Discord-Server

Backlog

Backlog

Estimated 9

This item hasn't started

WaterWords #254

Fixe Algolite

Client Enhancement Medium

WaterWords #255

Refactoring File Structure

Client Enhancement Medium

WaterWords #256

Server functionality Card Storm

Karten Server

WaterWords #257

Mirror Buildin

Karten Server

WaterWords #258

Server functionality Card Polymorph

Iteration 5 Karten Server

WaterWords #259

Server functionality Card Mass Mending

Iteration 7 Karten Server

WaterWords #260

Damaged Cards Doppelgänger available

Iteration 7 Karten Server

WaterWords #262

Water Animation for miss

Client Low UI

+ Add Item

Bugs

Estimated 3

WaterWords #265

server Crash - playing Thunder

Iteration 3 bug Server

WaterWords #276

Wrong placement for enemy ships

bug Client Medium UI

WaterWords #227

Vorbildschirm Fixen

Client Medium UI

WaterWords #202

Parallelize Dot inconsistent time

bug UI

WaterWords #232

CallWindCard: Handle moving into Ship

bug Client Medium Script Server

WaterWords #233

CallWindCard: Cell States Handling Client and Server

Client Medium Script Server

WaterWords #234

Make Healing Remove damaged Cells from GameShip Objects

bug Client Low Script UI

+ Add Item

ToDo

Estimated 2

This is actively being worked on

WaterWords #244

Time tracking - Nachrichten

Iteration 3 M Iteration 6 documentation High question

WaterWords #225

AbschlussVideo

Iteration 7

WaterWords #244

Präsentation Gliederung vorbereiten

Iteration 6

WaterWords #223

Doku fertigstellen

Iteration 6 Abschluss Präsentation

+ Add Item

In Progress

Estimated 3

WaterWords #241

Sounds

Iteration 1 M Client

WaterWords #232

Abschluss Präsentation

Iteration 7 Abschluss Präsentation

WaterWords #246

Server functionality Card Mass Mending

Iteration 7 Karten Server

+ Add Item

In Review

Estimated 5

WaterWords #239

Playing a card should cost Mana

Iteration 5 Iteration 6 Karten Server

WaterWords #232

Implement Wizard Assets

Iteration 7 Asset Client Medium UI Iteration 6

WaterWords #246

Server functionality Card Perfect Mending

Iteration 7 Karten Server

WaterWords #200

Server functionality Card Rise Sun

Iteration 5 Karten Server

WaterWords #248

Create Documentation from Docstrings

documentation Medium

+ Add Item

Done

Estimated 14

This has been completed

WaterWords #250

test-ToolsForMassCases

Iteration 14 UI Iteration 6 Test

WaterWords #222

delete Client

Iteration 6 Iteration 6

WaterWords #248

Refactor: Nicht-GemuteteKartenAusDeckLoeschen

Iteration 6 Karten Server

WaterWords #252

Cards Disapper from hand before card is done

Iteration 6 bug Client High Karten Iteration 6 Iteration 4

WaterWords #253

Cancel Cost of Card with Right Click

Iteration 6 Iteration 6 Client High Iteration 4

WaterWords #247

Feature-KartenVerschwindenAusHand

Iteration 6 Iteration 6 Client Karten Server Iteration 6 Iteration 4

WaterWords #244

GameState.cs Refactor

Iteration 6 Iteration 5 High Server Iteration 6

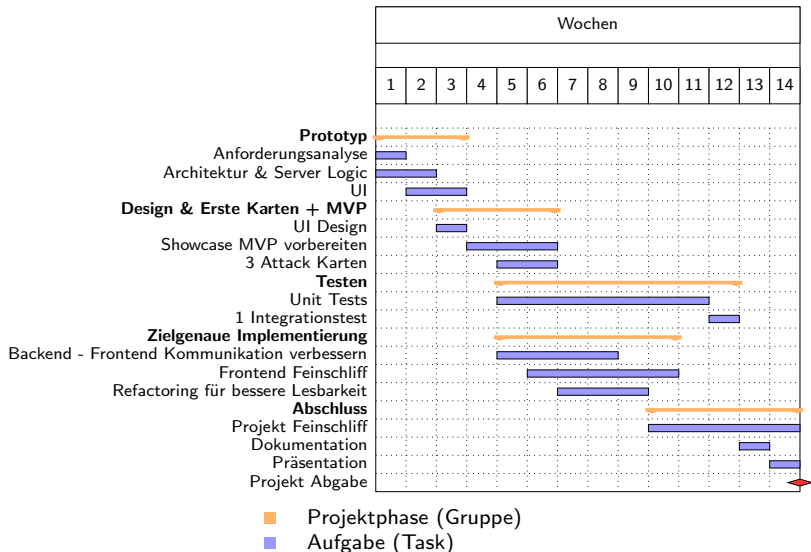
WaterWords #281

Gold needs to be incremented on the Server Side

Iteration 5 Iteration 5 High Server UI

+ Add Item

Projektplan



Rollen des Projektes

- ▶ Architektur: Justin Dewitz
- ▶ Dokumentation: Erick Zeiler
- ▶ Code-Qualität: Paul Schneider (abgesprungen)
- ▶ Git Repository: Max Kondratov

Ansprechpartner

Technologie und Architektur

- ▶ Programmiersprache: C#
- ▶ Raylib
- ▶ LiteNetLib
- ▶ Nuke für das Build-System
- ▶ CodeQL für die statische Code-Analyse
- ▶ GitHub Actions für die CD-Pipeline
- ▶ GitHub Pages für die Dokumentation
- ▶ Event-Driven Architektur
- ▶ Docker für die Containerisierung
- ▶ Hetzner Server für das Hosting

- ▶ Raylib für die grafische Darstellung
 1. Leichtgewichtige Library
 2. Einfacher Einstieg
 3. Kein Eigenes Key- und Maus-Event Handling

LiteNetLib für die Client-Server-Verbindung

1. Einfaches Setup?
2. Einfache Client-Server-Kommunikation über Nachrichten
3. Einfache Datenübertragung der Spieldaten

UI/Client



UI/Client

- ▶ Grafische Darstellung getrennt von Spielfunktionalität
 - ▶ Client übernimmt das Anzeigen und Handhabung des User Interfaces
- ▶ Gemeint sind Interaktive dargestellte Elemente, z.B.:
 - ▶ Das Main Menu, Die Kartenstapel, die Schiffe oder die Kartenhand
- ▶ Rendering und Client-Side Logik werden über Draw-Methoden direkt in der GameLoop ausgeführt wird.



UI-Beispiel aus der MainMenu Klasse

```
private static void HandleJoinButton(GameStateManager manager)
{
    Rectangle joinButton = new(
        (float)manager.screenWidth / 2 - 140,
        (float)manager.screenHeight / 2,
        246,
        72
    );

    bool hoverJoin = Raylib.CheckCollisionPointRec(Raylib.GetMousePosition(), joinButton);
    if (hoverJoin && Raylib.IsMouseButtonReleased(MouseButton.Left))
    {
        Raylib.PlaySound(SoundManager.ButtonSound);
        manager.SetStateToLobbyList();
    }

    Rectangle textureRec = new(0, 0, joinButtonAsset.Width, joinButtonAsset.Height);
    Raylib.DrawTexturePro(
        joinButtonAsset,
        textureRec,
        joinButton,
        Vector2.Zero,
        Of,
        Color.White
    );
    Raylib.DrawRectangleRec(joinButton, hoverJoin ? new(255, 255, 255, 31) : Color.Blank);
}
```

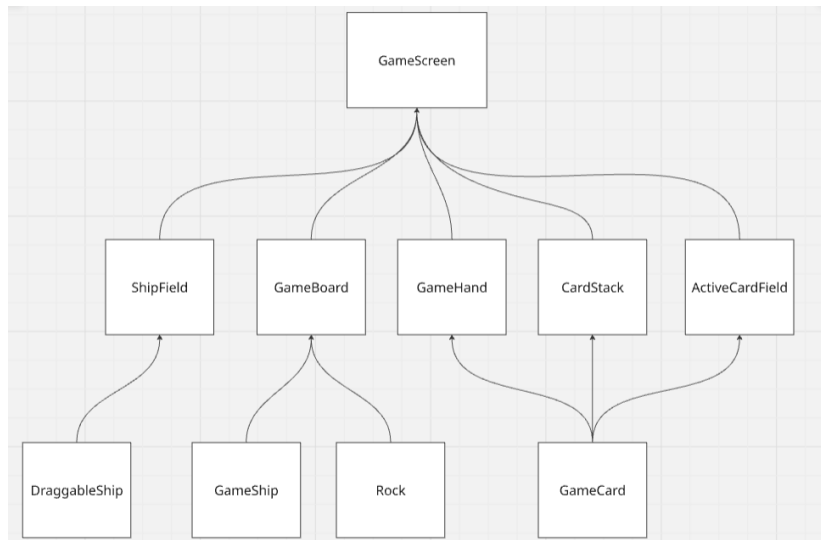
Nachrichtenempfang und Parsing im Client

```
private void HandleClientReceiveEvent(  
    NetPeer peer,  
    NetPacketReader reader,  
    byte channelNumber,  
    DeliveryMethod deliveryMethod  
)  
{  
    try  
    {  
        string messageType = reader.GetString();  
        Console.WriteLine($"[Client] Nachricht vom Server empfangen: {messageType}");  
  
        switch (messageType)  
        {  
            case "UpdatePauseState":  
                bool isPaused = reader.GetBool();  
                if (isPaused)  
                {  
                    GameStateManager.Instance.GetGamePauseManager().PauseGame();  
                }  
                else  
                {  
                    GameStateManager.Instance.GetGamePauseManager().ResumeGame();  
                }  
                break;  
            case "StartGame":  
                GameStateManager.Instance.SetStateToInGame();  
                break;  
            ...  
        }  
        ...  
    }  
}
```

Nachrichten senden an den Server

```
public void RequestCardBuy(string cardType)
{
    if (clientService.client != null && clientService.client.FirstPeer != null)
    {
        var writer = new NetDataWriter();
        writer.Put("BuyCard");
        writer.Put(cardType);
        clientService.client.FirstPeer.Send(writer, DeliveryMethod.ReliableOrdered);
        Console.WriteLine("[Client] Kaufe Karte");
    }
    else
    {
        Console.WriteLine("[Client] Kein Server verbunden, PlaceShip konnte nicht gesendet werden.");
    }
}
```


UI Struktur GameScreen



TextureManager

```
/// <summary>
/// Verwaltet das Laden und Entladen von Texturen für das Spiel.
/// </summary>
public class TextureManager
{
    private static List<Texture2D> textures = [];

    /// <summary>
    /// Lädt eine Textur aus einer Datei und speichert sie für späteres Entladen.
    /// </summary>
    /// <param name="file">Pfad zur Texturdatei</param>
    /// <returns>Die geladene Textur als <see cref="Texture2D"/></returns>
    public static Texture2D LoadTexture(string file)
    {
        var texture = Raylib.LoadTexture(file);
        textures.Add(texture);
        return texture;
    }

    /// <summary>
    /// Entlädt alle zuvor geladenen Texturen aus dem Speicher.
    /// </summary>
    public static void UnloadAllTextures()
    {
        foreach (var texture in textures)
        {
            Raylib.UnloadTexture(texture);
        }
    }
}
```

Server/Backend

- ▶ Das Backend ist in C# mit der Library LiteNetLib geschrieben
- ▶ Ein globaler Server der eine Lobby auf Hetzner bereitstellt
- ▶ Server wird in Docker-Containern ausgeführt
- ▶ Der Server wird auf dem Port 7777/UDP bereitgestellt

Backend - Client Kommunikation

Event-Driven Architektur

- ▶ UDP für die Echtzeit-Kommunikation
- ▶ Nachrichten basiertes Protokoll für beidseitige Kommunikation

```
Wireshark Client 1
[Client] Spieler 1 hat nun 5 Gold.
[Client] Nachricht vom Server empfangen: UpdateMana
[Client] Spieler 1 hat nun 3 Mana.
[Client] Nachricht vom Server empfangen: UpdateGold
[Client] Spieler 1 hat nun 6 Gold.
[Client] Nachricht vom Server empfangen: UpdateGold
[Client] Spieler 1 hat nun 7 Gold.
[Client] Nachricht vom Server empfangen: UpdateMana
[Client] Spieler 1 hat nun 4 Mana.
[Client] Nachricht vom Server empfangen: UpdateGold
[Client] Spieler 1 hat nun 8 Gold.
[Client] Nachricht vom Server empfangen: UpdateGold
[Client] Spieler 1 hat nun 9 Gold.
[Client] Nachricht vom Server empfangen: UpdateMana
[Client] Spieler 1 hat nun 10 Gold.
[Client] Nachricht vom Server empfangen: UpdateGold
[Client] Spieler 1 hat nun 11 Gold.
[Client] Nachricht vom Server empfangen: UpdateMana
[Client] Spieler 1 hat nun 5 Mana.
[Client] Nachricht vom Server empfangen: UpdateGold
[Client] Spieler 1 hat nun 12 Gold.
[Client] Nachricht vom Server empfangen: UpdateMana
[Client] Spieler 1 hat nun 6 Mana.
[Client] Nachricht vom Server empfangen: UpdateGold
[Client] Spieler 1 hat nun 13 Gold.
[Client] Nachricht vom Server empfangen: UpdateGold
[Client] Spieler 1 hat nun 7 Mana.
[Client] Nachricht vom Server empfangen: UpdateMana
[Client] Spieler 1 hat nun 14 Gold.
[Client] Nachricht vom Server empfangen: UpdateGold
[Client] Spieler 1 hat nun 8 Gold.

Wireshark Server
[GoldHandler] Player 1 Gold +1 (Neuer Stand: 5)
[GoldHandler] Player 2 Gold +1 (Neuer Stand: 5)
[ManaHandler] Player 1 Mana +1 (Neuer Stand: 15)
[ManaHandler] Player 2 Mana +1 (Neuer Stand: 15)
[GoldHandler] Player 1 Gold +1 (Neuer Stand: 6)
[GoldHandler] Player 2 Gold +1 (Neuer Stand: 6)
[GoldHandler] Player 1 Gold +1 (Neuer Stand: 7)
[GoldHandler] Player 2 Gold +1 (Neuer Stand: 7)
[ManaHandler] Player 1 Mana +1 (Neuer Stand: 6)
[ManaHandler] Player 2 Mana +1 (Neuer Stand: 6)
[GoldHandler] Player 1 Gold +1 (Neuer Stand: 8)
[GoldHandler] Player 2 Gold +1 (Neuer Stand: 8)
[GoldHandler] Player 1 Gold +1 (Neuer Stand: 9)
[GoldHandler] Player 2 Gold +1 (Neuer Stand: 9)
[GoldHandler] Player 1 Gold +1 (Neuer Stand: 10)
[GoldHandler] Player 2 Gold +1 (Neuer Stand: 10)
[GoldHandler] Player 1 Gold +1 (Neuer Stand: 11)
[GoldHandler] Player 2 Gold +1 (Neuer Stand: 11)
[ManaHandler] Player 1 Mana +1 (Neuer Stand: 16)
[ManaHandler] Player 2 Mana +1 (Neuer Stand: 16)
[GoldHandler] Player 1 Gold +1 (Neuer Stand: 12)
[GoldHandler] Player 2 Gold +1 (Neuer Stand: 12)
[ManaHandler] Player 1 Mana +1 (Neuer Stand: 17)
[ManaHandler] Player 2 Mana +1 (Neuer Stand: 17)
[GoldHandler] Player 1 Gold +1 (Neuer Stand: 13)
[GoldHandler] Player 2 Gold +1 (Neuer Stand: 13)
[ManaHandler] Player 1 Mana +1 (Neuer Stand: 18)
[ManaHandler] Player 2 Mana +1 (Neuer Stand: 18)
[GoldHandler] Player 1 Gold +1 (Neuer Stand: 14)
[GoldHandler] Player 2 Gold +1 (Neuer Stand: 14)
```

Spielzustand / GameState

- ▶ Programm wird in Zustands-Klassen (States) beschrieben
- ▶ Spiel wird in der GameState-Klasse beschrieben
- ▶ Spielinstanzen können getrennt werden

Beispiel:

```
public void CheckGameOver()
{
    foreach (var player in players)
    {
        if (player != null && ShipHandler.AreAllShipsDestroyed(player))
        {
            var winner = players.FirstOrDefault(p => p != player);
            if (winner != null)
            {
                BroadcastGameOver(winner, player);
            }
        }
    }
}
```

Factory Pattern

- ▶ Das Factory-Pattern wird für die Erstellung von Kartenobjekten verwendet
- ▶ Für jede Kartenart gibt es eine eigene Factory

Beispiel: DamageCardFactory

```
public static class DamageCardFactory
{
    public static IDamageCard Create(CardVariant variant)
    {
        return variant switch
        {
            CardVariant.Firebolt => new FireboltCard(),
            CardVariant.FrostBolt => new FrostBoltCard(),
            CardVariant.Lightning => new LightningCard(),
            _ => throw new ArgumentException($"Unknown variant: {variant}")
        };
    }
}
```

Vorteile:

- ▶ Neue Karten können einfach ergänzt werden
- ▶ Spiellogik bleibt unverändert
- ▶ Klare Trennung von Erstellung und Verwendung

Shared

- ▶ Shared enthält alle Klassen, die sowohl im Client als auch im Server verwendet werden
- ▶ Definitionen der Karten und der Kartentypen
- ▶ Definitionen der Schiffe und der Schiffs-Typen
- ▶ Definition von Mana

Beispiel: Cards

Definition:

```
// Damage Variants
{ CardVariant.MagicAttack, CardType.Damage },
{ CardVariant.ArcaneMissile, CardType.Damage },
{ CardVariant.Firebolt, CardType.Damage },
{ CardVariant.Fireball, CardType.Damage },
{ CardVariant.GreedHit, CardType.Damage },
{ CardVariant.FrostBolt, CardType.Damage },
{ CardVariant.LifeSteal, CardType.Damage },
//[...]
```

Nutzung:

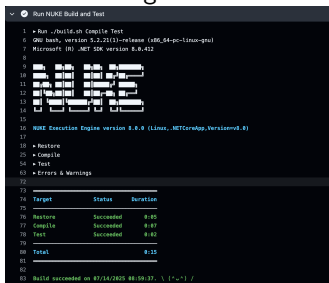
```
public void HandleCast(Cards card, GameBoard.Point hoveredCoords)
{
    if (clientService.client != null && clientService.client.FirstPeer != null)
    {
        NetDataWriter writer = new();
        writer.Put("CastCard");
        writer.Put(card.Variant.ToString());
        writer.Put(hoveredCoords.X);
        writer.Put(hoveredCoords.Y);
        clientService.client.FirstPeer.Send(writer, DeliveryMethod.ReliableOrdered);
        Console.WriteLine($"[Client] Karte wirken: {card.Variant} an Position ({hoveredCoords
            .X}, {hoveredCoords.Y})");
    }
    else
    {
        //[...]
    }
}
```


- ▶ CI-Pipeline
- ▶ CD-Pipeline
- ▶ Statische Code-Analyse
- ▶ Pull Requests mit 4-Augen Prinzip
- ▶ Dokumentation auf Github Pages

CI/CD Pipeline

► CI

- Nuke build für die Continuous-Integration
- dotnet *restore*, *build*, *test*, werden bei jedem PR ausgeführt

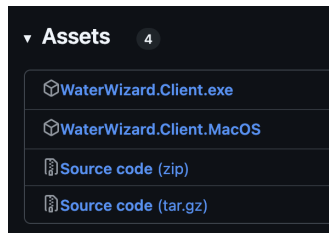


```
1 ► Run ./build.sh Compile Test
6 One Build, version 5.2.2(1)-release (x86_64-pc-linux-gnu)
7 Microsoft (R) .NET SDK version 8.0.402
8
9
10
11
12
13
14
15
16 Nuke Execution Engine version 3.0.0 (Linux, .NETCoreApp,Version=v8.0)
17
18 ► Restore
25 ► Compile
34 ► Test
63 ► Errors & Warnings
72
73
74
75
76
77
78
79
80
81
82
83 Build succeeded on 07/14/2025 08:50:37. \ [!+!]
```

Target	Status	Duration
Restore	Succeeded	0:05
Compile	Succeeded	0:07
Test	Succeeded	0:02
Total		0:15

► CD

- Github Actions für das Continuous-Deployment
- Baut eine exe Datei für Windows und eine MacOS App



CI-Pipeline Konfiguration

```
name: NUKE Build CI

on:
  push:
    branches: [ "main", "dev" ]
  pull_request:
    branches: [ "**" ]

workflow_dispatch:

jobs:
  build:
    runs-on: ubuntu-latest

    permissions:
      actions: read
      contents: read
      security-events: write

    steps:
      - name: Checkout repository
        uses: actions/checkout@v4
        with:
          fetch-depth: 0

      - name: Setup .NET SDK
        uses: actions/setup-dotnet@v4
        with:
          dotnet-version: '8.0.x'

      - name: Run NUKE Build and Test
        run: ./build.sh Compile Test
```

CD-Pipeline Konfiguration (Teil 1)

```
- name: Publish Client (Windows)
  run: dotnet publish src/WaterWizard.Client/WaterWizard.Client.csproj -c Release -r win-x64 --
        self-contained true -o ./publish/win --verbosity normal

- name: Publish Server (Windows)
  run: dotnet publish src/WaterWizard.Server/WaterWizard.Server.csproj -c Release -r win-x64 --
        self-contained true -o ./publish/win-server

- name: Publish Client (MacOS)
  run: dotnet publish src/WaterWizard.Client/WaterWizard.Client.csproj -c Release -r osx-x64 --
        self-contained true -o ./publish/osx --verbosity normal

- name: Publish Server (MacOS)
  run: dotnet publish src/WaterWizard.Server/WaterWizard.Server.csproj -c Release -r osx-x64 --
        self-contained true -o ./publish/osx-server

- name: Prepare release assets
  run: |
    mkdir release-assets

    # Find the actual executable names
    WIN_EXE=$(find ./publish/win -name "*.exe" -type f | head -1)
    OSX_EXE=$(find ./publish/osx -type f -executable | grep -v "\.dll$" | grep -v "\.so$" | head
              -1)
```

CD-Pipeline Konfiguration (Teil 2)

```
if [ -n "$WIN_EXE" ]; then
  cp "$WIN_EXE" release-assets/WaterWizard.Client.exe
else
  echo "Warning: No Windows executable found"
fi

if [ -n "$OSX_EXE" ]; then
  cp "$OSX_EXE" release-assets/WaterWizard.Client.MacOS
else
  echo "Warning: No MacOS executable found"
fi

echo "Release assets:"
ls -la release-assets/

- name: Create GitHub Release & Upload Assets
  uses: softprops/action-gh-release@v2
  with:
    tag_name: ${ steps.get_version.outputs.tag }
    name: Release ${ steps.get_version.outputs.version }
    body: ${ steps.changelog.outputs.changelog }
    draft: false
    prerelease: false
    files: release-assets/*
  env:
    GITHUB_TOKEN: ${ secrets.GITHUB_TOKEN }
```

Statische Code-Analyse

- ▶ CodeQL für die statische Code-Analyse
- ▶ CodeQL eigene Konfigurationsdatei
 - ▶ Möglich auch in der CI Konfigurationsdatei
 - ▶ Best Practice getrennt
 - ▶ Simplere Konfiguration durch separate Datei
- ▶ Ergebnisse in GitHub-Security

CodeQL Konfiguration (Teil 1)

```
name: CodeQL

on:
  push:
    branches: [ main, dev ]
  pull_request:
    branches: [ '**' ]
  workflow_dispatch:

jobs:
  analyze:
    name: Analyze
    runs-on: ubuntu-latest
    permissions:
      security-events: write
      actions: read
      contents: read

    strategy:
      fail-fast: false
    matrix:
      language: ['csharp']

    steps:
      - name: Checkout repository
        uses: actions/checkout@v4
        with:
          fetch-depth: 0
```

CodeQL Konfiguration (Teil 2)

```
- name: Setup .NET SDK
  uses: actions/setup-dotnet@v4
  with:
    dotnet-version: '8.0.x'

- name: Initialize CodeQL
  uses: github/codeql-action/init@v3
  with:
    languages: ${ matrix.language }
    config-file: .github/codeql/codeql.yml

- name: Build with NUKE (required by CodeQL)
  run: ./build.sh Compile

- name: Perform CodeQL Analysis
  uses: github/codeql-action/analyze@v3
  with:
    category: '/language:${ matrix.language }'
```


Containerisierung

- ▶ Docker für die Containerisierung des Servers
- ▶ Dockerfile im Server-Verzeichnis
- ▶ Docker Compose im root-Verzeichnis

Dockerfile Code

Dockerfile

```
FROM mcr.microsoft.com/dotnet/sdk:8.0 AS build
WORKDIR /source

COPY WaterWizards.sln .

COPY src/WaterWizard.Server/WaterWizard.Server.csproj ./src/WaterWizard.Server/
COPY src/WaterWizard.Shared/WaterWizard.Shared.csproj ./src/WaterWizard.Shared/
COPY src/WaterWizard.Client/WaterWizard.Client.csproj ./src/WaterWizard.Client/
COPY src/WaterWizardTests/WaterWizardTests.csproj ./src/WaterWizardTests/

RUN dotnet restore WaterWizards.sln

COPY src/WaterWizard.Server/ ./src/WaterWizard.Server/
COPY src/WaterWizard.Shared/ ./src/WaterWizard.Shared/
COPY src/WaterWizard.Client/ ./src/WaterWizard.Client/
COPY src/WaterWizardTests/ ./src/WaterWizardTests/

RUN dotnet publish src/WaterWizard.Server/WaterWizard.Server.csproj -c Release -o /app/
publish

FROM mcr.microsoft.com/dotnet/runtime:8.0 AS final
WORKDIR /app

COPY --from=build /app/publish .

EXPOSE 7777/udp

ENTRYPOINT ["dotnet", "WaterWizard.Server.dll"]
```

Docker-Compose Code

docker-compose

```
version: '3.8'

services:
  waterwizard-server:
    build:
      context: .
      dockerfile: ./src/WaterWizard.Server/Dockerfile
    ports:
      - "7777:7777/udp"
    environment:
      - PUBLIC_ADDRESS=${SERVER_IP}
```

Technische Erfahrungen

Analyse

Backlog 8 Estimate: 0 ...

This item hasn't been started

 WaterWizards #254

Finale Abgabe

Orga

 finale Abgabe

 WaterWizards #155 

Refactoring File Structure


Client

enhancement


Medium

 WaterWizards #98

Bug

 Water
server C

P1

 Water
Wrong p

bug

 Water
Vollbild

Erfahrungen und Fazit

- ▶ Team Mitglied hat das Team verlassen.
- ▶ Seine Arbeit konnte übernommen werden.
- ▶ Gesamte Arbeitslast ist gestiegen.