

# 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

Client Server

WaterWords #257

Mirror Buildin

Client Server

WaterWords #258

Server functionality Card Polymorph

Iteration 5 Client Server

WaterWords #259

Server functionality Card Mass Mending

Iteration 7 Client Server

WaterWords #260

Damaged Cards Doppelgänger available

Iteration 7 Client 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 Medium Documentation

WaterWords #225

AbschlussVideo

Iteration 7

WaterWords #244

Präsentation Gliederung vorbereiten

Iteration 8

WaterWords #223

Doku fertigstellen

Iteration 8 Abschluss Präsentation

+ Add Item

In Progress

Estimated 3

WaterWords #241

Sounds

Iteration 1 Medium Client

WaterWords #232

Abschluss Präsentation

Iteration 8 Abschluss Präsentation

WaterWords #246

Server functionality Card Mass Mending

Iteration 7 Client Server

+ Add Item

In Review

Estimated 5

WaterWords #239

Playing a card should cost Mana

Iteration 5 Iteration 6 Karten Server

WaterWords #248

Implement Wizard Assets

Iteration 7 Asset Client

WaterWords #246

Server functionality Card Perfect Mending

Iteration 8 Karten Server

WaterWords #200

Server functionality Card Rise Sun

Iteration 8 Karten Server

WaterWords #248

Create Documentation from Docstrings

Documentation Medium

+ Add Item

Done

Estimated 14

This has been completed

WaterWords #240

test-ToolsForMassCases

Iteration 14 Iteration 6 Test

WaterWords #222

delete Client

Iteration 6 Iteration 8

WaterWords #248

Refactor: Nicht-GemuteteKartenAusDeck/Löschen

Iteration 8 Karten

WaterWords #233

Cards Disappear from hand before card is done

Iteration 8 Bug Client

WaterWords #254

Cancel Cost of Card with Right Click

Iteration 8 Client High

WaterWords #247

Feature-KartenverschwindenAusHand

Iteration 8 Karten

WaterWords #244

GameState.cs Refactor

Iteration 5 High Server

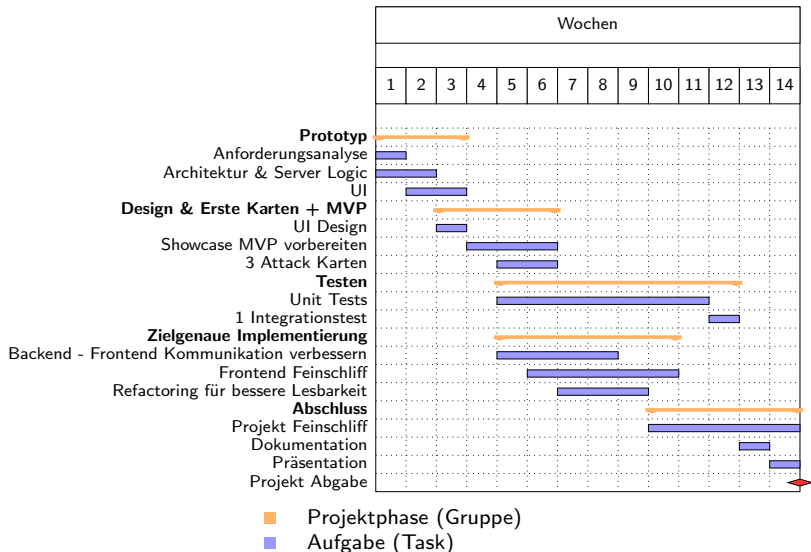
WaterWords #281

Gold needs to be incremented on the Server Side

Iteration 5 High Server

+ 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 für die grafische Darstellung
- ▶ LiteNetLib für die Client-Server-Verbindung
- ▶ 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
- ▶ Grafische Darstellung getrennt von Spielfunktionalität:
  - ▶ Client übernimmt das Anzeigen und Handhabung des User Interfaces
- ▶ Gemeint sind Interaktive dargestellte Elemente, z.B.:
  - ▶ Die Kartenstapel, die Schiffe oder die Kartenhand
- ▶ Alle Elemente werden durch eine Methode Draw angezeigt, welche in der GameLoop ausgeführt wird.
- ▶ Die wenige Logik, die auf dem Client ausgeführt wird, wird in diesen Draw Methoden ausgeführt

# Nachrichtenempfang + 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;  
            ...  
        }  
        ...  
    }  
}
```

# Spiel-Bildschirm / GameScreen



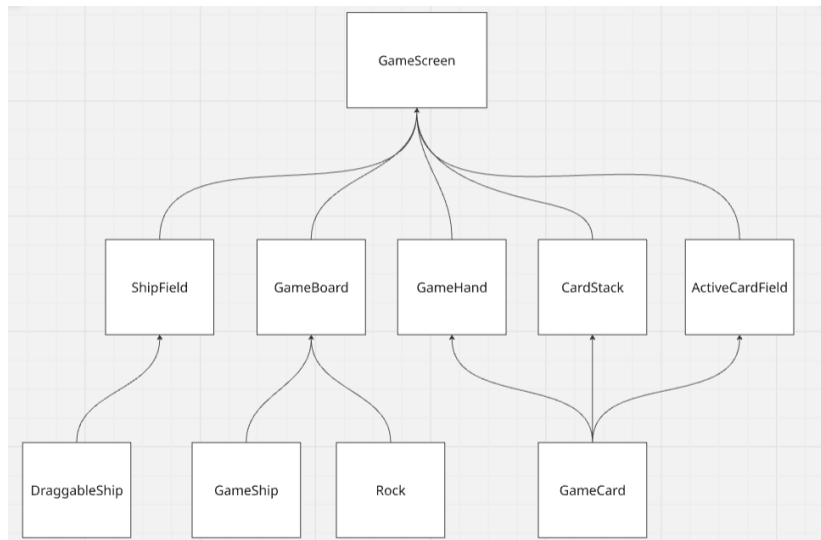
# UI-Beispiel aus der GameHand Klasse

```
public virtual void Draw(bool front)
{
    int availableHandWidth = (int)(ScreenWidth * 0.2f);
    int totalCardWidth = Cards.Count * CardWidth;
    int excess = totalCardWidth - availableHandWidth;
    int offset = excess > 0 ? excess / Cards.Count : 0;
    for (int i = 0; i < Cards.Count; i++)
    {
        //[...]

        Cards[i].Draw(centralX + cardX, cardY, front);

        DrawCardPreview(front, i, cardX, effectiveCardWidth);
    }
}
```

# 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

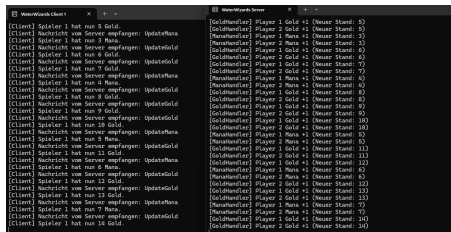
- ▶ Das Backend ist in C# mit der Library LiteNetLib geschrieben
- ▶ Ein globaler Server der eine Lobby auf Hetzner bereitstellt
- ▶ Server wird in Docker-Containern ausgeführt
- ▶ Der Server wird auf dem Port 7777/UDP bereitgestellt



# Backend - Client Kommunikation

## Event-Driven Architektur

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



The image shows two terminal windows side-by-side. The left window, titled 'WunderWach Client', displays a series of messages from the client to the server, such as 'Spieler 1 hat nun 5 Gold.' and 'Nachricht vom Server empfangen: UpdateMana'. The right window, titled 'WunderWach Server', displays the corresponding server responses, such as '[GoldHandler] Player 1 Gold +1 (Neuer Stand: 5)' and '[ManaHandler] Player 1 Mana +1 (Neuer Stand: 15)'. The messages show a sequence of game events and state updates for two players.

- ▶ Programm wird in Zustands-Klassen (States) beschrieben
- ▶ Spiel wird in der GameState-Klasse beschrieben

## Ausschnitt:

```
public class GameState
{
    public NetPeer[] players = new NetPeer[2];
    public static readonly int boardWidth = 12;
    public static readonly int boardHeight = 10;
    public readonly Cell[,] boards = new Cell[2][,];
    public Cell[,] Player1 => boards[0];
    public Cell[,] Player2 => boards[1];
    public readonly List<Cards>[] hands;
    private readonly NetManager server;
    public readonly ServerGameStateManager manager;
    public List<Cards> Player1Hand => hands[0];
    public List<Cards> Player2Hand => hands[1];
    public static List<Cards>? ActiveCards //[...]
    public static List<Cards>? UtilityStack //[...]
```

# 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

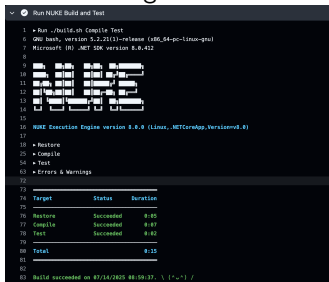
- ▶ Shared enthält alle Klassen, die sowohl im Client als auch im Server verwendet werden
- ▶ Enthält die Definitionen der Karten und der Kartentypen
- ▶ Enthält die Definitionen der Schiffe und der Schiffs-Typen

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

# CI/CD Pipeline

## ► CI

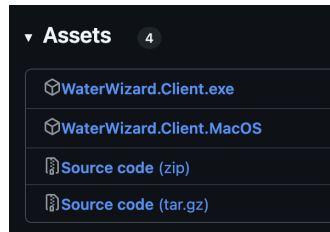
- Nuke build für die Continuous-Integration
- dotnet *restore*, *build*, *test*, werden bei jedem PR ausgeführt



```
1 > Run ./build.sh Compile Test
2
3 One Build, version 5.2.2(1)-release (x86_64-pc-linux-gnu)
4 Microsoft (R) .NET SDK version 8.0.402
5
6
7
8
9
10
11
12
13
14
15
16 Nuke Execution Engine version 1.0.0 (Linux, .NETCoreApp,Version=v8.0)
17
18 > Restore
19
20 > Compile
21
22 > Test
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

## ► 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