# Dokumentation

Fachinformatiker Anwendungsentwicklung

# Thema

Tower Defense


| | |
|---|---|
| Prüfungsteilnehmer: | Ricardo Liebig |
| Betrieb: | cbm GmbH Bremen<br>Wegesende 3-4<br>28195 Bremen |
| Projektbetreuer: | Sven Lilienthal |
| Durchführungszeitraum: | 12.05.2025 bis 25.05.2025 |

# Inhaltsverzeichnis

# 1 Einleitung

Das Ziel dieses Projekts war die Entwicklung eines einfachen Tower-Defense-Spiels unter Verwendung der Programmiersprache Java und der Swing-Bibliothek für die grafische Benutzeroberfläche. Es sollte eine funktionale Basisversion eines Tower-Defense-Spiels erstellt werden, das es dem Spieler ermöglicht, Türme zu platzieren, Gegnerwellen abzuwehren und das Spielziel zu erreichen. Alle Projektphasen, von der Konzeption über die Entwicklung bis hin zu den Tests, wurden von einer Person durchgeführt. Die Wahl von Java und Swing als Technologien ermöglichte eine effiziente Entwicklung und die Nutzung von bereits vorhandenen Kenntnissen.

# 2 Projektbeschreibung

## 2.1 Projektumfeld

Das Projekt wurde auf einem Windows-PC in einer lokalen Entwicklungsumgebung umgesetzt. Für die Entwicklung kam Java 11 in Verbindung mit Swing zum Einsatz. Als IDE wurde IntelliJ IDEA verwendet, um eine effiziente und produktive Entwicklungsumgebung zu gewährleisten. Die grafische Benutzeroberfläche wurde vollständig mit Swing gestaltet, was eine einfache Handhabung und schnelle Anpassung der Benutzeroberfläche ermöglichte.

## 2.2 Organisatorische Vorgaben

Das Projekt wurde im Zeitraum vom 12.05.2025 bis 23.05.2025 durchgeführt und durfte 80 Arbeitsstunden nicht überschreiten.

## 2.3 Ziel des Projektes

Ziel dieses Projekts war die Entwicklung eines Tower-Defense-Spiels, bei dem die grundlegenden Mechaniken eines Tower-Defense-Genres umgesetzt werden. Dabei sollten neben der Platzierung von Türmen und dem Abwehren von Gegnerwellen auch die Ergebnisse in einer SQL-Datenbank gespeichert werden. Der Fokus lag auf der Integration der Highscore-Funktionalität, bei der nach jedem Spiel (Sieg oder Niederlage) der Spielname und der Highscore gespeichert werden, um den Spielern eine Rückmeldung über ihre Leistungen zu geben.

## 2.4 Projektabgrenzung

Einzelne Handlungsschritte im Detail sind aus zeitlichen Gründen nicht Bestandteil dieses Projekts. Der Fokus liegt auf der Konzeption und Umsetzung der Kernfunktionen eines Tower-Defense-Spiels sowie der technischen Einbindung einer Highscore-Speicherung über eine SQL-Datenbank. Die folgende Liste gibt einen Überblick über den groben Projektablauf und zeigt, welche Punkte Bestandteil des Projekts waren und welche nicht:

- Entwurf und technische Richtlinien für das Spielsystem definieren
- Aufbau der Swing-Oberfläche zur Interaktion mit dem Spiel
- Implementierung der Spiellogik mit Bewegung, Angriff und Ressourcenverwaltung
- Anbindung einer SQL-Datenbank zur Highscore-Speicherung
- Implementierung der Benutzerinteraktion
- Durchführung eines Testdurchlaufs inklusive Datenbankeintrag
- Weitere Features wie Level-Auswahl, dynamische Gegnerpfade oder Online-Ranking sind nicht Bestandteil des Projekts

Damit bleibt das Projekt in einem klar umrissenen Rahmen. Es liefert eine funktionierende Basisversion, die bei Bedarf in zukünftigen Arbeitsschritten erweitert werden kann.

### 2.5 Projektschnittstellen

Eine zentrale Schnittstelle im Projekt ist die SQL-Datenbank, die mit der Java-Anwendung über JDBC kommuniziert. Nach jedem abgeschlossenen Spiel wird entweder der Spielname und der Highscore bei einem Sieg oder einer Niederlage in der Datenbank gespeichert. Die Datenbankinteraktionen umfassen das Hinzufügen von Datensätzen sowie das Abrufen und Anzeigen des Highscores. Dies geschieht über vorbereitete SQL-Abfragen, die durch die Anwendung ausgelöst werden.

# 3 Projektplanung

## 3.1 Ist-Analyse

Die Ist-Analyse beschreibt die Ausgangssituation vor Beginn des Projekts. Es lagen keine bestehenden Projektstrukturen, Quellcodes oder vorbereiteten Module vor. Die Entwicklungsumgebung musste vollständig eingerichtet werden, einschließlich:

- Installation von Java 11 SDK
- Einbindung von Swing
- Installation und Konfiguration der IDE
- Einrichtung eines lokalen MySQL-Servers zur Speicherung von Spielergebnissen
- Konfiguration eines einfachen Datenbank-Schemas mit einer Tabelle zur Speicherung von Spielername und Punktestand

Die Netzwerkanbindung diente ausschließlich für Recherche- und Updatezwecke. Das Projekt selbst lief vollständig lokal ohne externe Abhängigkeiten.

### 3.1.1 Netzwerkplan Ist-Zustand

Da es sich um ein rein lokal durchgeführtes Einzelprojekt handelte, existierte keine strukturierte Netzwerkinfrastruktur. Die gesamte Entwicklungs- und Laufzeitumgebung lief auf einem einzigen Rechner. Eine Datenbankverbindung bestand lediglich lokal zur eigenen MySQL-Instanz. Es gab keine Verbindungen zu externen Servern oder Netzwerken.

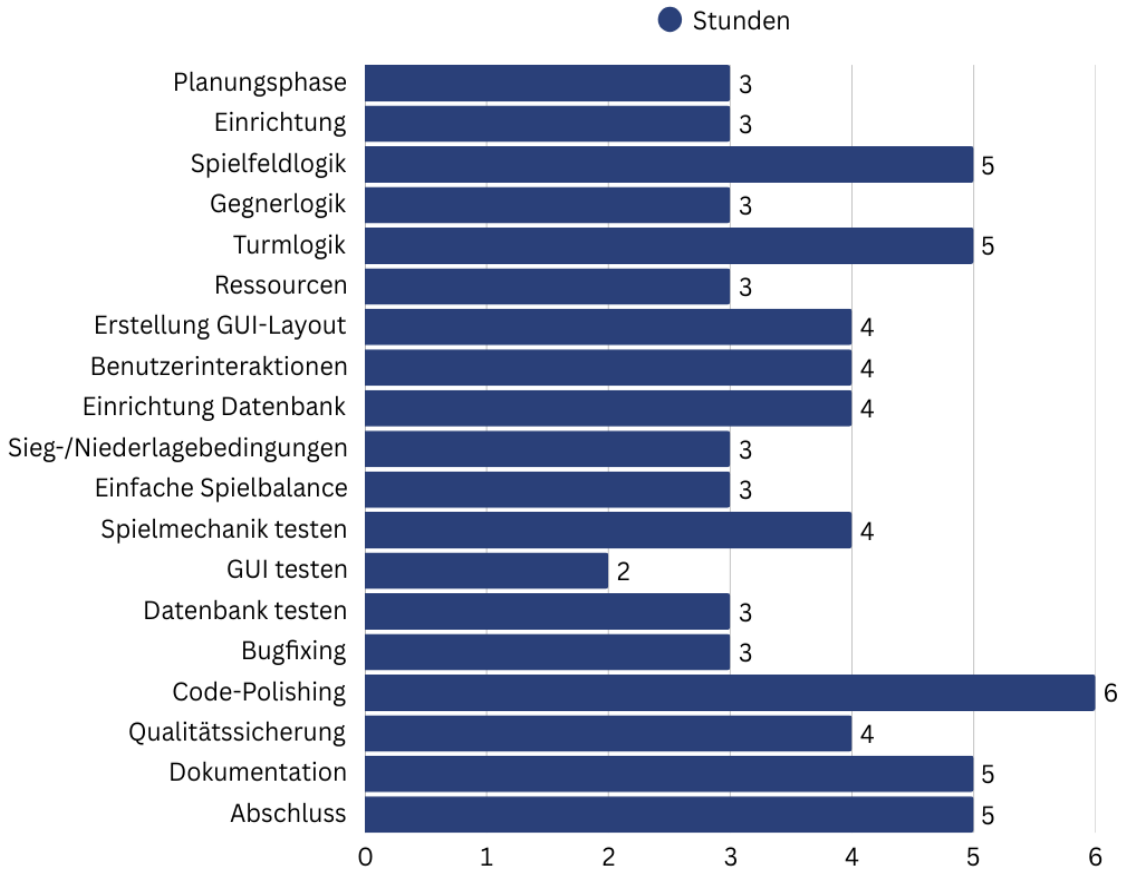
Ein Netzwerkplan im klassischen Sinne war daher nicht erforderlich.

Für zukünftige Erweiterungen, beispielsweise zur Bereitstellung eines Online-Highscores oder eines Multiplayer-Modus, müsste das Projekt um entsprechende Netzwerkschnittstellen ergänzt werden.

## 3.2 Soll-Planung

Im Rahmen der Soll-Planung wurde definiert, wie das Tower-Defense-Spiel technisch aufgebaut sein soll, welche Spielfunktionen umzusetzen sind und welche Struktur die Anwendung erhalten soll. Der Schwerpunkt lag auf der Realisierung eines funktionalen Spielflusses, bei dem Gegner in Wellen erscheinen und vom Spieler strategisch platzierte Türme abgewehrt werden. Das Spiel sollte modular aufgebaut sein, um spätere Erweiterungen zu erleichtern.

## 3.3 Projektablaufplan



## 3.4 Personalplanung

Das Projekt wurde vollständig allein durchgeführt. Es gab keine Aufgabenverteilung im klassischen Sinne. Alle Rollen wie Planung, Entwicklung, Testing und Dokumentation wurden von einer Person übernommen. Die Personalplanung war somit sehr übersichtlich:

| Rolle | Zuständigkeit |
|---|---|
| Projektleitung | Zeitplanung, Projektstrukturierung, Zieldefinition |
| Softwareentwicklung | Implementierung der Anwendung mit Java und Swing |
| Datenbankverwaltung | Einrichtung der SQL-Datenbank und Anbindung via JDBC |
| Test-/ Qualitätssicherung | Funktionstests, Fehleranalyse, Korrekturmaßnahmen |
| Dokumentation | Erstellung dieser Projektdokumentation |

## 3.5 Personalkosten

| Ressource | Aufwand | Stundensatz | Kosten |
|---|---|---|---|
| Personal(80 Std.) | 80h | 25€/h | 2000 € |
| Softwarelizenzen | – | – | 0 € |
| Hardware (Abschreibung) | – | – | 100 € |
| Gesamt | | | 2100 € |

## 3.6 Nutzwertanalyse

Im Rahmen des Projekts wurde eine Nutzwertanalyse durchgeführt, um die geeignetste Programmiersprache für die Entwicklung des Tower-Defense-Spiels zu identifizieren. Zur Auswahl standen Java, Python und JavaScript, wobei besonderes Augenmerk auf Eigenschaften gelegt wurde, die für Spieleentwicklung, insbesondere im 2D-Bereich sowie auf eine saubere Architektur und Erweiterbarkeit entscheidend sind.

### 3.6.1 Bewertungskriterien

Folgende Bewertungskriterien wurden festgelegt und gewichtet:

| Kriterium | Gewichtung |
|---|---|
| Einfachheit der Datenbankanbindung | 10% |

| | |
|---|---|
| Performance für 2D-Spiele | 20% |
| Strukturierbarkeit / OOP | 30% |
| Cross-Plattform-Fähigkeit | 10% |
| Einarbeitungszeit | 5% |
| GUI-/Grafikunterstützung | 20% |
| IDE-Unterstützung | 10% |
| **Gesamt** | **100%** |

### 3.6.2 Bewertung der Technologien

Die drei Technologien wurden anhand der obigen Kriterien mit Punktzahlen von 1 (sehr schlecht) bis 10 (sehr gut) bewertet:

| Kriterium | Gewichtung | Java | Python | JS |
|---|---|---|---|---|
| Einfachheit der Datenbankanbindung | 10% | 9 | 7 | 6 |
| Performance für 2D-Spiele | 20% | 6 | 4 | 5 |
| Strukturierbarkeit / OOP | 30% | 8 | 6 | 5 |
| Cross-Plattform-Fähigkeit | 10% | 8 | 7 | 9 |
| Einarbeitungszeit | 5% | 5 | 9 | 8 |
| GUI-/Grafikunterstützung | 20% | 6 | 5 | 6 |
| IDE-Unterstützung | 5% | 6 | 6 | 6 |
| **Gesamtpunkte** | 100% | **7,05** | **5,75** | **5,70** |

### 3.6.3 Ergebnis und Entscheidung

Die Nutzwertanalyse zeigt, dass Java mit einem Gesamtnutzwert von 7,05 deutlich vor Python (5,75) und JavaScript (5,70) liegt. Java überzeugt vor allem in den Bereichen Performance, Objektorientierung Cross-Plattform-Fähigkeit und GUI-/Grafikunterstützung. Aufgrund dieser Bewertung fiel die Entscheidung, das Tower-Defense-Spiel mit Java zu realisieren.

Die Auswahl bietet nicht nur eine saubere Architektur mit objektorientierten Prinzipien, sondern auch eine gute Performance und Erweiterbarkeit für künftige Features wie Highscore-Tabellen oder Mehrspieler-Modi.

# 4 Realisierung

## 4.1 Technische Umsetzung

Die Umsetzung des Spiels erfolgte in mehreren modularen Entwicklungsschritten. Der Fokus lag auf der Trennung von Spiellogik, grafischer Oberfläche und Datenbankanbindung, um eine wartbare und erweiterbare Codebasis zu schaffen. Die Anwendung wurde in Java entwickelt und verwendet Swing für die Benutzeroberfläche sowie JDBC für die Verbindung zur SQL-Datenbank.

Die Hauptmodule des Spiels umfassen:

- Spielsteuerung: Zentrale Steuerung der Spielmechanik
- Spielfeld: Repräsentation des Spielfelds mit logischer Rasterstruktur, Platzierungslogik für Türme und Pfadverfolgung für Gegner.
- Turm-Logik: Verwaltung der Türme inklusive Angriffsreichweite, Nachladezeit.
- Gegner-Logik: Bewegung entlang eines vordefinierten Pfads, Lebenspunkte und Geschwindigkeit.
- Benutzeroberfläche**:** Gestaltung des Layouts mithilfe von Swing, Einbindung von Buttons zur Turmauswahl.
- Datenbank-Modul: Einfache Klasse zur Verwaltung von Verbindungen, Einfügen von Highscores und Abrufen von Daten zur Anzeige im Spiel.

## 4.2 Spielmechanik

Zu Beginn des Spiels startet der Spieler mit einer festen Menge an Ressourcen. Diese können genutzt werden, um Türme an vorgegebenen Positionen zu platzieren. Die Türme haben eine eigene Angriffsgeschwindigkeit und Reichweite. Die Gegner erscheinen in einer Welle und bewegen sich automatisch entlang eines festgelegten Pfads von einem Start- zu einem Zielpunkt. Wird ein Gegner nicht rechtzeitig eliminiert, erreicht er das Ziel und das Spiel wird beendet. Sobald die

Lebenspunkte der Gegner auf null sinken, ist das Spiel verloren. Wird eine Welle erfolgreich abgewehrt, wird das Spiel beendet und der Spieler kann seinen Namen zum Highscore eintragen.

## 4.3 Manuelles Testing

Da im Rahmen dieses Projekts keine automatisierten Unit-Tests durchgeführt wurden, wurde das Testing hauptsächlich durch manuelles Testen realisiert. Dabei wurden alle wesentlichen Spielmechaniken, die Benutzeroberfläche sowie die Interaktion mit der Anwendung manuell überprüft, um sicherzustellen, dass das Spiel wie erwartet funktioniert und benutzerfreundlich ist.

### 4.3.1 Testziele

Die Hauptziele des manuellen Testens waren:

- **Funktionalität prüfen:** Sicherstellen, dass alle Spielmechaniken (z. B. Turmbau, Gegnerplatzierung, Ressourcenverwaltung) ordnungsgemäß funktionieren.
- **Benutzeroberfläche testen:** Überprüfung der Benutzerfreundlichkeit und Lesbarkeit der GUI, Testen der Interaktivität.
- **Fehlerfreiheit:** Identifikation und Behebung von visuellen oder funktionalen Fehlern, wie z. B. falsche Berechnungen.

### 4.3.2 Durchführung der Tests

Die Tests wurden in mehreren Iterationen durchgeführt:

| Testphase | Erfolgreich? |
|---|---|
| Datenbankerstellung | Ja |
| Datenbankanbindung | Ja |
| Gegner, Türme mit Indikator, Projektile zeichnen | Ja |
| Lebensbalken in der Gegner Klasse | Ja |
| Gegner-Leben wird korrekt berechnet und nimmt Schaden | Ja |
| Spiel wird korrekt geupdatet | Ja |
| move()-Funktion von Gegner und Projektil | Ja |
| Spielereingaben werden korrekt verarbeitet | Ja |
| GUI-Benutzeroberfläche | Ja |

### 4.3.3 Fazit des manuellen Testens

Das manuelle Testen ermöglichte es, viele Aspekte des Spiels gründlich zu überprüfen und zu verbessern. Zwar konnte durch diese Methode keine vollständige Testabdeckung wie bei automatisierten Tests erreicht werden, jedoch konnten kritische Fehler frühzeitig erkannt und behoben werden. Das manuelle Testen hat sich als effektives Mittel erwiesen, um die Funktionalität, Benutzerfreundlichkeit und Stabilität des Tower-Defense-Spiels sicherzustellen.

# 5 Grafische Benutzeroberfläche

Die grafische Benutzeroberfläche (GUI) stellt den zentralen Zugangspunkt für die Interaktion der Spieler mit dem Tower-Defense-Spiel dar. Sie wurde benutzerfreundlich, übersichtlich und modular aufgebaut, um sowohl eine einfache Bedienung als auch eine klare Trennung der Zuständigkeiten im Code zu gewährleisten.

## 5.1 Aufbau und Elemente

Die GUI wurde mit Java Swing umgesetzt. Sie gliedert sich in mehrere Hauptkomponenten:

- **Spielanzeige:**
  Zeigt das Spielfeld, Ressourcen des Spielers (z. B. Leben, Punkte),

sowie Platzierung von Türmen

→ Klassen: TowerDefense, GamePanel, Tower, Enemy

- **Spielende:**
  Nach Spielende (Sieg oder Niederlage) erscheint ein Dialogfenster mit der erreichten Punktzahl und der Option, seinen Namen einzutragen, das Spiel erneut zu spielen oder das Spiel zu beenden.
  → Klassen: TowerDefense, GamePanel, DatabaseManager

### 5.2 Technische Umsetzung

Die GUI-Komponenten wurden nach dem Model-View-Controller (MVC)-Prinzip entworfen:

- Model: Verwaltung des Spielzustands (z. B. Spiellogik, Gegner, Tower)
- View: Darstellung der Oberfläche (Swing-Komponenten)
- Controller: Verarbeitung der Benutzereingaben (Mausklicks)

Dies ermöglicht eine saubere Trennung von Darstellung und Logik sowie eine leichtere Erweiterbarkeit (z. B. neue Türme).

## 6 Qualitätssicherung

Zur Sicherstellung der Qualität des entwickelten Tower-Defense-Spiels wurden während des gesamten Projektverlaufs verschiedene Maßnahmen getroffen. Obwohl keine automatisierten Tests zum Einsatz kamen, wurde besonderer Wert auf Stabilität, Benutzerfreundlichkeit und Wartbarkeit gelegt. Die Qualitätssicherung lässt sich in mehrere Teilbereiche gliedern.

### 6.1 Projektstruktur und Codequalität

Bereits bei der Planung wurde darauf geachtet, das Projekt in klar getrennte Komponenten zu unterteilen. Die Spiellogik, Benutzeroberfläche und Datenverarbeitung wurden in logisch abgeschlossenen Klassen und Paketen organisiert. Diese modulare Struktur erleichterte nicht nur die Entwicklung, sondern auch die spätere Fehlersuche und Erweiterung.

Im Verlauf der Umsetzung wurde der Quellcode regelmäßig überprüft und überarbeitet. Dabei lag der Fokus auf Lesbarkeit, Wiederverwendbarkeit und sauberer Methodengestaltung. Die Einhaltung objektorientierter Prinzipien und die Verwendung aussagekräftiger Bezeichner sorgten für eine gute Wartbarkeit und Verständlichkeit des Programmcodes.

## 6.2 Benutzerfreundlichkeit

Ein weiterer wichtiger Aspekt der Qualitätssicherung war die Gestaltung der Benutzeroberfläche. Diese wurde bewusst übersichtlich und intuitiv gehalten, sodass sich auch unerfahrene Nutzer schnell zurechtfinden können. Einzelne Elemente wie Buttons, Statusanzeigen oder Auswahlfelder wurden mehrfach angepasst, um eine möglichst klare Bedienung zu gewährleisten. Auch externes Feedback spielte hier eine Rolle. Testpersonen äußerten Rückmeldungen zur Verständlichkeit der Oberfläche, zur Platzierung von Bedienelementen und zur Übersichtlichkeit. Diese Hinweise wurden genutzt, um gezielte Verbesserungen vorzunehmen.

## 6.3 Performance und Stabilität

Die technische Stabilität des Spiels war ein zentrales Qualitätsziel. Besonders bei hoher Spiellast, etwa bei vielen gleichzeitig aktiven Gegnern und Türmen, wurde die Performance beobachtet. Durch Optimierungen in den Zeichenroutinen und eine effiziente Ereignisverarbeitung konnte das Spiel auch in solchen Situationen flüssig laufen.

# 7 Projektabschluss

## 7.1 Übergabe

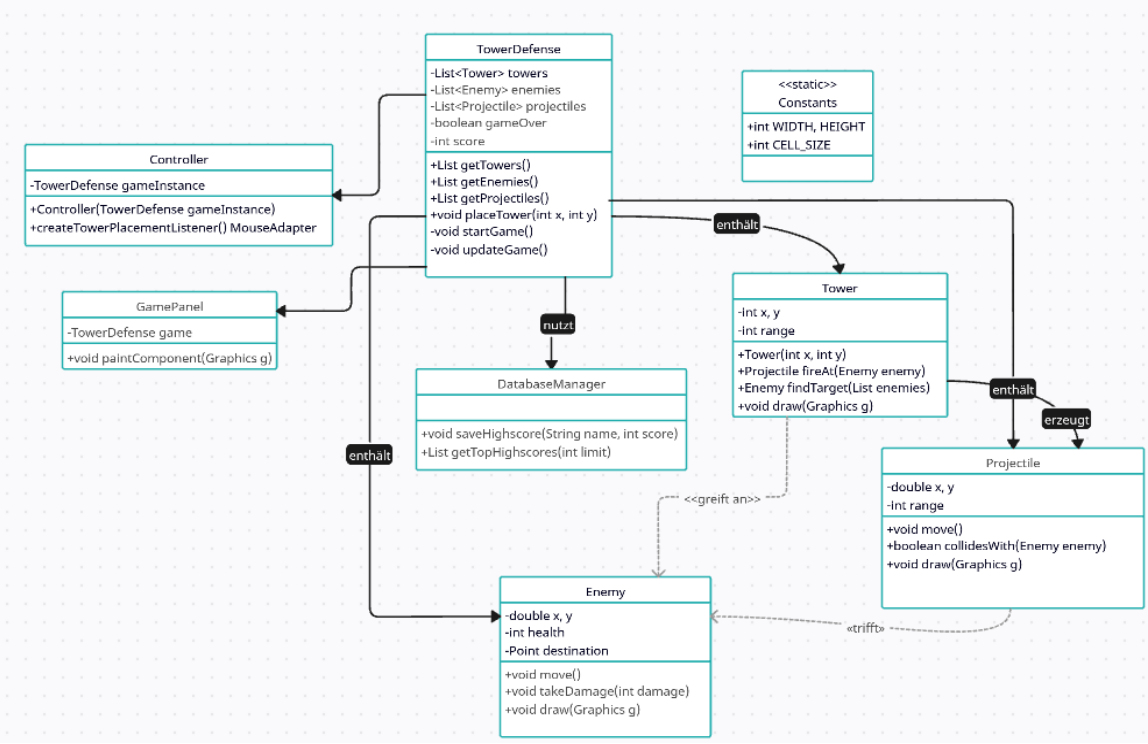
Am 25.05.2025 fand die Übergabe an Herrn Lilienthal über GitHub statt. So ist es dem Dozenten möglich gewesen das Projekt ausgiebig zu prüfen.
Auf diesem Wege konnte man die wichtigsten funktionalen und nicht-funktionalen Anforderungen überblicken und testen.

## 7.2 Fazit und Ausblick

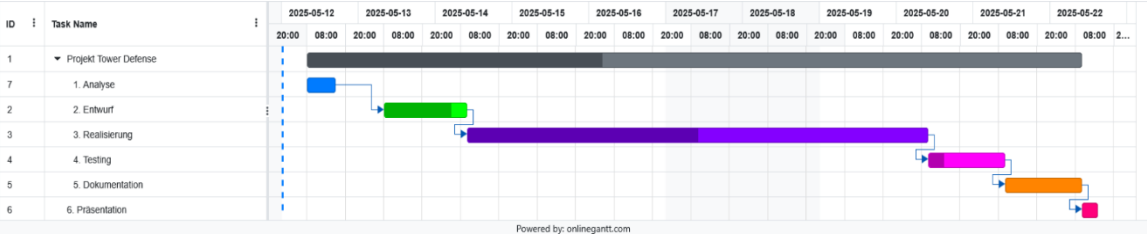
Das Projekt wurde erfolgreich abgeschlossen. Trotz Abweichungen bei den Ist-Zeiten war es möglich die angegebene Bearbeitungszeit von 80 Stunden einzuhalten. Dabei konnte ich viele Erfahrungen im Umgang mit Java und SQL-Datenbanken sammeln.

# A Anhang

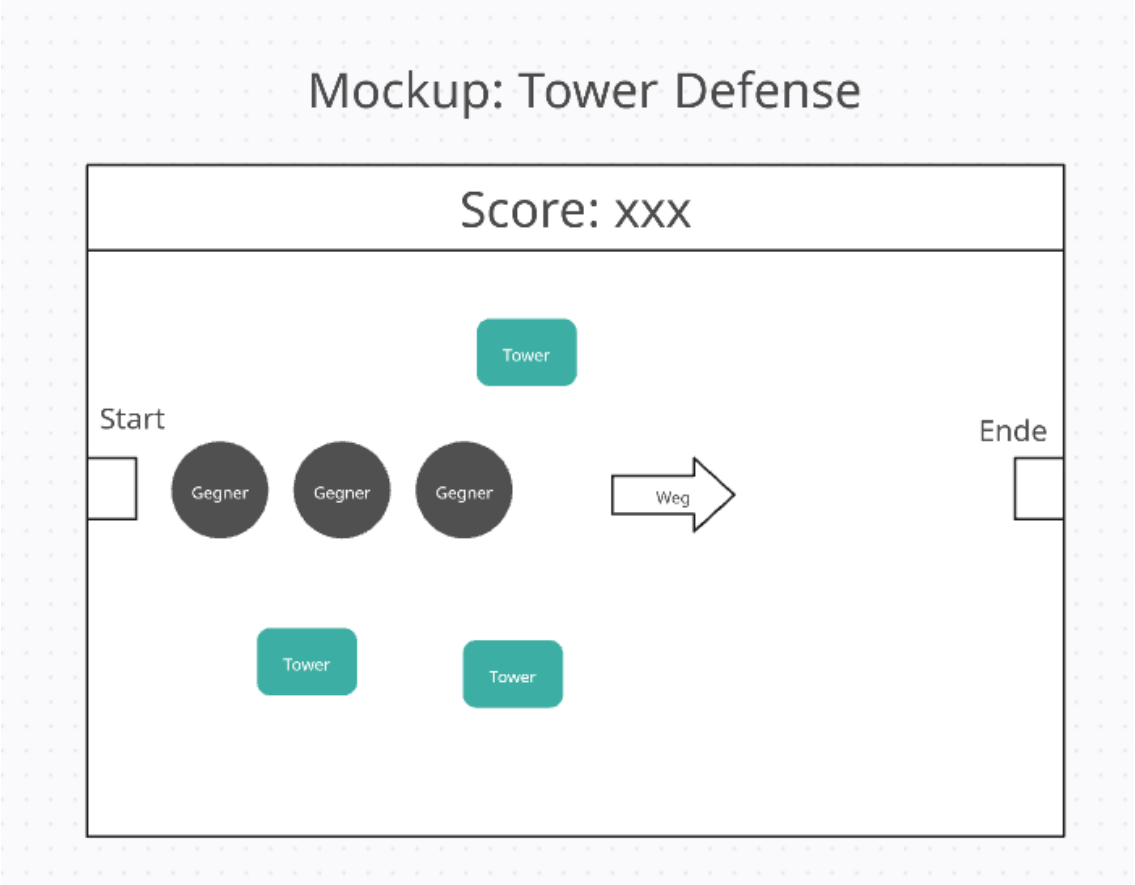
## A.1 UML-Klassendiagramm




## A.2 Gantt Diagramm



## A.3 Mockup


Mockup: Tower Defense
Score: xxx
Tower
Start
Ende
Gegner
Gegner
Gegner
Weg
Tower
Tower