Bei dem Vorgehensmodell haben wir uns wegen folgender Gründe für das Wasserfallmodell entschieden **Begründung**:

- Klare Anforderungen zu Beginn ohne erwartbare Änderungen
- strukturiertes Vorgehen mit klaren Zielen zu Zieldatum
- erhöhte Wahrscheinlichkeit der Zielerreichung
- Kompatibilität mit Vorlesungsaufbau

Pflichtenheft

1. Zielbestimmung

Bei dem Produkt handelt es sich um ein elektronisches Spiel, welches auf dem Brettspiel Quoridor basiert. Das Spiel soll bis zu vier Personen erlauben, Quorridor lokal auf demselben Gerät mit einer graphischen Oberflächee zu spielen. Ein Online Multiplayer ist nicht vorgesehen. Das Produkt soll lediglich die essenziellen Features eines Walking Skeleton zum Spielen von Quorridor enthalten. Ein Vorteil gegenüber einem herkömmlichen Brettspiel ist die Verfügbarkeit, da es downloadbar für den PC überall zur Verfügung steht.

2. Einsatz

Zielgruppen:

Brettspielspieler jeglicher Altersgruppen mit Affinität zu digitalen Spielformat. Die Personas sind diesem Pflichtenheft beigefügt.

Einsatzbereiche:

Das Spiel soll für den freizeitlichen Gebrauch entworfen werden. Es soll kompetitiv gestaltet sein, sodass Spieler ihre Fähigkeiten vergleichen können, sowie den Spieler eine angenehme Möglichkeit bieten sich die Zeit zu vertreiben. Es soll die Konzentration und strategisches Denken verbessern.

Szenarien:

Das Produkt könnte beispielsweise durch eine Familie genutzt werden, die zuhause oder auf einer Reise zusammen etwas spielen möchte. Ein weiteres Szenario ist ein Spiel von Studenten in der Hochschule. Auch Zuhause bei Spieleabenden kann das Spiel zum Einsatz kommen, da dann nicht die Brettspielversion gekauft werden muss und digitale Hilfen bereit gestellt werden.

3. Umgebung

Das Spiel wird für Desktop-PCs und Laptops unter den gängigen Betriebssystemen Windows und MacOS entwickelt. Es handelt sich dabei um eine eigenständige Anwendung. Ein UML-Klassendiagramm, UML-Sequenzdiagramm sowie ein UML-Zustandsdiagramm sind diesem Pflichtenheft beigefügt. Ebenso ist eine High-Level-Architektur als Diagramm angehangen. Für eine Modulspezifikation dient das UML-Klassendiagramm. Es können die beigefügten Entwurfsmuster verwendet werden. Die Begründungen für die Anwendung dieser Muster befinden sich im Code.

4. Funktionalität

- Angabe Benutzername, um die Anzahl der Siege anzuzeigen
- Anzeige der Spielregeln in einer kurzen verständlichen Form
- Anzeige des Spielbretts mit unterschiedlich gefärbten Spielfiguren und Wänden, Anzeige der Startseite jeder Spielfigur
- Anzeige der Anzahl an noch verfügbaren Wände für einen Spieler
- Anzeige des Spielers, der am Spielzug ist
- Möglichkeit in einem Spielzug eine Wand mittels Mausklick zu setzen oder seine Figur mit Mausklick zu ziehen
- Sicherstellen, dass Wände richtig gesetzt werden
- Sicherstellen, dass Figuren richtig ziehen können und gegebenenfalls übereinander um die Ecke springen können
- Sicherstellen, dass Figuren immer einen Weg auf die gegenüberliegende Seite besitzen

- Möglichkeit, einen Zug rückgängig zu machen
- Benachrichtigung über den Sieger

5. Daten

Daten, die Verarbeitet werden sind ein vom Nutzer freiwillig erstellbares Spielerprofil, sowie die Informationen über das aktuelle Spiel. Ein Spielerprofil enthält dabei nur einen vom Nutzer ausgedachten Nutzernamen und die Anzahl an bereits gewonnenen Spielen, jedoch keine personenbezogenen Daten wie E-Mail, Vorname, Nachname oder Alter. Die Informationen sind nicht gesondert durch ein Passwort oder ähnliches geschützt und die Spieldaten werden auch nicht verschlüsselt gespeichert. Die Speicherung der Daten wird mit einer lokalen SQLite Datenbank in einer Datei auf der Festplatte des Computers umgesetzt. Ein Impressum ist nicht notwendig. Das ER-Modell ist dem Pflichtenheft beigefügt.

6. Leistungen

Um eine angenehme Nutzererfahrung zu bieten werden folgende Leistungsanforderungen aufgestellt: - **Skalierbarkeit**: Die Spieleranzahl soll nicht über 4 Erhöht werden können. Es soll technisch möglich sein in Zukunft die Spielfeldgröße sowie die Zugmöglichkeiten auf eine Größe von maximal 15 mal 15 zu erweitern, ohne dass andere Leistungsanforderungen nicht mehr eingehalten werden. Die maximale Anzahl an Spielerprofilen soll dabei mindestens 100 betragen, wobei der benötigte Speicherplatz nicht 100 MB überschreiten darf.

- **Durchsatz**: Wegen der geringen Datenmenge werden keine besonderen Anforderungen an den Durchsatz gestellt.
- **Antwortzeiten**: Die Latenz, wenn eine Aktion durch den Nutzer ausgeführt wurde, darf auf einem M1 MacBook Pro mit 16 GB RAM auf MacOS 14.4 maximal eine halbe Sekunde betragen. Dies vom gilt vom Klick des Nutzers bis zum Anzeigen auf dem Bildschirm, wobei die Latenz des Bildschirms, der Maus und der Tastatur nicht beachtet werden.
- **Ladezeiten**: Die Initiale Ladezeit des Programms bis zum Anzeigen auf dem Bildschirm darf auf einem M1 MacBook Pro mit 16 GB RAM auf MacOS 14.4 maximal eine Sekunde dauern, wobei die Latenz des Bildschirms, der Maus und der Tastatur nicht beachtet werden.

7. Benutzeroberfläche

Das System sollte den Spielern jederzeit den aktuellen Status des Spiels in einer minimalistischen Abbildung der Realität mitteilen, dies betrifft den Status des Spielfelds und die Position der Figuren. Der Nutzer kann das Spiel mittels Mausklicks navigieren, wobei nur grundsätzlich valide Spielzüge graphisch angezeigt werden. Spielzüge die die Regel, dass ein offener Weg für jeden Spieler bleiben muss, dürfen angezeigt werden, sollten bei dem Versuch der Durchführung jedoch eine Fehlermeldung “Zug nicht erlaubt. Jeder Spieler muss einen offenen Weg auf die andere Seite haben” ausgeben. Der Spieler kann das Spiel nicht speichern. Es soll eine abgestimmte Auswahl an Farben und Symbolen genutzt werden, um dem Spieler Informationen zu vermitteln. Das User-Interface soll fehlerhafte Aktionen verhindern, oder den Spieler auf diese hinweisen. Das Spiel soll für Einsteiger und erfahrene Nutzer gleichermaßen intuitiv nutzbar sein. Um dies um zusetzen soll die Benutzeroberfläche minimalistisch und frei von nicht essenziellen Elementen sein. Um es Anfängern einfacher zu machen, sollte das Spiel eine Beschreibung der Regeln enthalten. Ein grundsätzliches Mockup mit den wichtigsten Komponenten ist dem Pflichtenheft beigelegt.

8. Qualitätsziele

- **Wartbarkeit**: Die Wartbarkeit und Erweiterbarkeit der Software soll einfach möglich sein.
- **Zuverlässigkeit**: Das Spiel soll zuverlässig und ohne Fehler laufen.

- **Sicherheit**: Es gibt keine Qualitätsziele für die Sicherheit Um die Einhaltung der Qualitätsziele sicherzustellen, wird die Software mithilfe von Unit-Test hinreichend getestet.

9. Entwicklungsumgebung

- **Softwaretools**: Für das Schreiben des Codes wird Neovim und Visual Studio Code genutzt. Für die Versionierung Git, Github und Github Desktop.
- **Programmierrichtlinien**: Für die Qualität des Codes ist es essenziell, dass sich an gemeinsame Code-Standards gehalten wird. Für die Naming-Conventions werden die Standard Rust-Naming-Conventions für Rust und die camelCase Naming-Conventions für Svelte genutzt. Eine Statement-Test-Coverage der Logik von 50% wird erreicht.
- **Versionskontrolle**: Die Versionierung wird mit Git und Github umgesetzt. Dabei wird jedes Feature auf einem eigenen Branch entwickelt, welcher nach dem Merge mit dem Main-Branch gelöscht wieder gelöscht wird. Der Merge wird mit Github Pull Requests durchgeführt wobei vor dem Merge ein anderer Entwickler den Code geprüft haben muss.

10. Erweiterungen

10.1 Spielregeln

Quoridor ist ein Brettspiel, dass mit zwei oder vier Personen gespielt werden kann. Klassischerweise ist das Spielbrett ein 9x9-Feld. Jeder Spieler startet mit seiner Spielfigur auf dem Mittelfeld einer Randspalte. Zusätzlich erhält jeder Spieler 10 Wände der länge 2 Spielfelder (bei 4 Spielern 6 Wände). Das Ziel ist es, die gegenüberliegende Randseite zu erreichen. Dafür ziehen die Spieler abwechselnd, beziehungsweise im Uhzeigersinn. Ein Zug beinhält entweder eine Bewegung der eigenen Spielfigur um eins in einer geraden Spielfeldrichtung (nicht schräg) oder das Setzen einer Wand zwischen 4 Felder. Eine Wand kann nicht durch eine Spielfigur passiert werden. Eine Wand kann nicht eine andere Wand schneiden. Auch dürfen Wände nicht so gesetzt werden, dass einer Spielfigur es nicht mehr möglich ist, die andere Seite zu erreichen. Es gewinnt, wessen Spielfigur als erstes die andere Seite erreicht.

User Story Map

User activities
Features

Benutzername angeben | Spielübersicht | Spiel spielen | Quoridor verwalten

Features

User stories
Release 1/Prio 1

		Als DominoDoris62 möchte ich gerne eine kurze prägnante Übersicht über die Spielregeln in Deutsch, damit ich auch verstehe, was das Ziel ist und jederzeit nachschauen kann, welche Züge legal sind.		Als Magnus Charlson möchte ich, dass die Regeln von Quoridor (insb. dass der Weg nicht versperrt werden darf) korrekt implementiert sind, damit niemand schummeln kann oder plötzlich komische Dinge passieren	Als DominoDoris62 möchte ich gerne an einem Gerät eine Quoridor Partie gegen meine Freundin spielen, wobei immer abwechselnd gezogen wird, weil mir das sehr viel Spaß macht, wenn sie vorbei schaut	Als Ferris Rusty möchte ich einen einfachen Aufbau der Applikation mit kommentierten und dokumentierten Code, um mir den Wartungsprozess so einfach wie möglich zu machen	Als Ferris Rusty möchte ich das Backend der Applikation in Rust entwickeln, da Rust modern, sicher und hardwarenah ist.
				Als DominoDoris62 möchte ich gerne grafisch sehen, wo welche Spielfigur steht und welche Wände gesetzt werden sowie wie viele Wände ich noch setzen darf, damit ich einen bessere Spielübersicht habe	Als DominoDoris62 möchte ich, dass die Spielfiguren sich klar vom Hintergrund und voneinander farblich abheben, um zu erkennen wo wer ist.		
				Als Magnus Charlson möchte ich nach einem Spiel die Benachrichtigung erhalten, wer gewonnen oder verloren hat, um mich bei einem Sieg zu freuen.			

Walking Skeleton

Release 2/Prio 2

Als Magnus Charlson möchte ich gerne einen Benutzernamen ohne Passwort angeben, damit ich auch in mehreren Sessions meine Anzahl an Siegen anzeigen lassen kann	Als Ferris Rusty möchte ich Benuternamen händisch aus der Datenbank löschen können, da ich so vermeiden kann, dass zu viele Benuternamen existieren	Als Magnus Charlson möchte ich gerne die Spielübersicht in Englisch, da ich kaum Deutsch verstehe.	Als DominoDoris62 möchte ich gerne einen Link zu den vollständigen Spielregeln auf Wikipedia um die Regeln besser zu verstehen, falls mir die Kurzfassung nicht ausreicht.	Als DominoDoris62 möchte ich gerne an einem Gerät eine Quoridor Partie gegen meinen 3 Freundinen zu 4. spielen, wobei immer abwechselnd gezogen wird, weil mir das sehr viel Spaß macht, zusammen zu spielen	Als Magnus Charlson möchte ich gerne gegen einen Bot spielen, um meine Fähigkeiten zu verbessern, falls gerade kein Mitspieler da ist		
				Als Magnus Charlson möchte ich gerne sehen, wie viele Spiele mein Benutzername bereits gewonnen hat, weil ich mich dadurch besser fühle.	Als DominoDoris62 möchte ich die möglichen Züge meiner Spielfigur vorgeschlagen bekommen und durch Mausklick auf die richtige Stelle auswählen		
				Als DominoDoris62 möchte ich gerne einen Zurück-Button falls ich mich verklickt habe.	Als DominoDoris62 möchte ich gerne die Wände durch einen Mausklick auf die entsprechende Stelle mit zuvor durch drüber hovern markierter Position erhalten, um die Wände wie gewünscht zu setzen.		

Release 3/Prio 3

Als Ferris Rusty möchte ich, dass Benutzer, dessen letztes Spiel mehr als 6 Monate zurück liegt automatisch aus der Datenbank gelöscht werden, um ein unnötiges Ansammeln zu vermeiden.				Als Magnus Charlson möchte ich gerne einen Online-Spiel Modus um gegen beliebige andere Spieler anzutreten			

Personas

Name: Ferris Rusty
Age: 61
Nationalität: amerikanisch
Location: Thailand
Stand: verheiratet

Beruf:

• Senior Rust Developer
• geht auf alle Gaming Messen verkleidet als Krabbe

Hobbies / Interessen

Rewrite Linux in Rust

Ziele:

Dauer-Urlaub in Thailand
App läuft

Wünsche:

High Performance
Wenig Wartungsaufwand
Einfache Architektur

Frustration:

Schlecht dokumentierter Code
unnötig komplizierter Code

Stärken:

Kernel Programmierung
Userorientiertes Arbeiten

Schwächen:

No Weaknesses

Name: Magnus Charlson
Age: 24
Nationalität: Norwegisch
Ort: Tønsberg, Norwegen
Stand: ledig

Beruf:

studiert Theoretische Physik

Hobbies / Interessen

spielt Schach seit er 5 ist
spielt auch gerne andere Strategiespiele
spielt am liebsten Blitzschach
hat 1800 ELO bei LiChess

Ziele:

Kognitive Weiterentwicklung

Wünsche:

spannende Vorlesungsbeschäftigung mit seinen Kommillitonen
Gewinnen durch Training

Frustration:

Lange Wartezeiten

Stärken:

Schach
Technik
Logisches Denken

Schwächen:

Verlieren
Keine Gedult

Name: DominoDoris62
Age: 81
Ort: Lederhose
Stand: verwitwet

Beruf:

Rentnerin

Hobbies / Interessen

Spieleabend

Ziele:

Zeit mit Freunden Verbringen

Wünsche:

Einfaches Design
Neues Spiel für den Spiele Abend

Frustration:

Komplizierte Bedinung

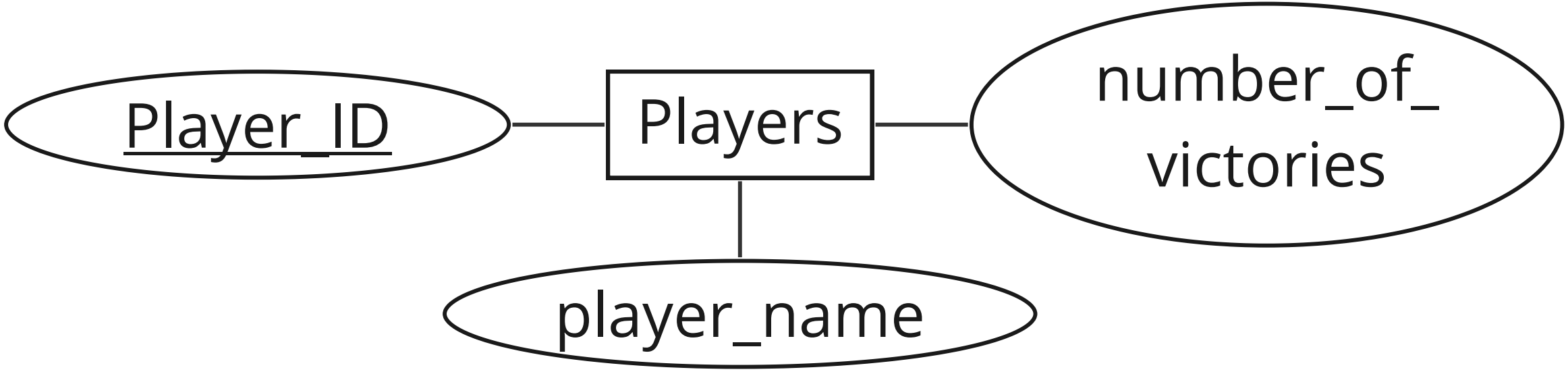
Stärken:

Schafkopf
Dame

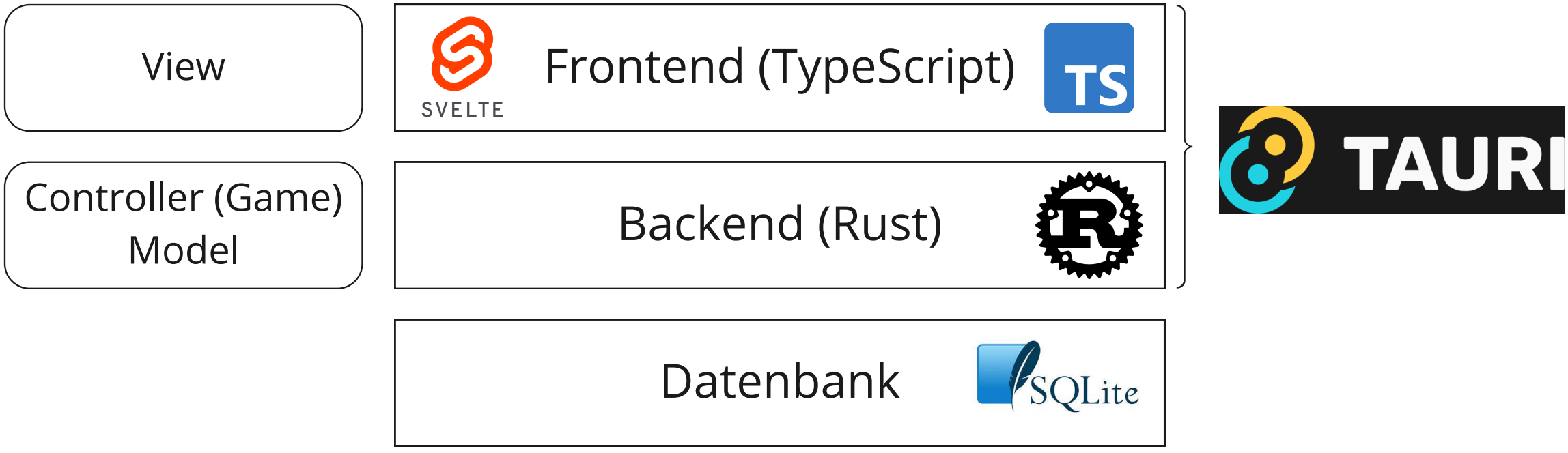
Schwächen:

Umgang mit Computern

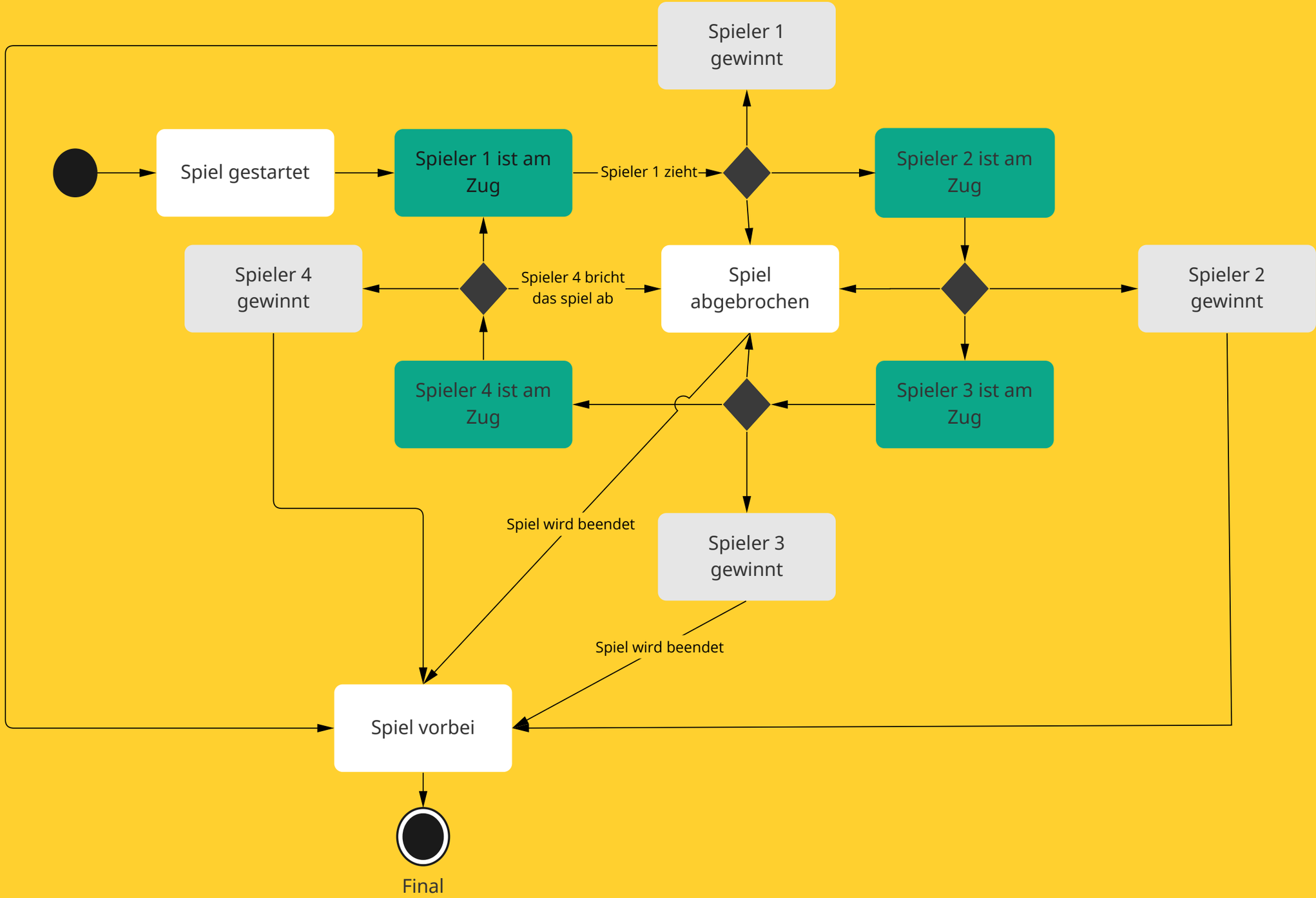
ER-Diagramm



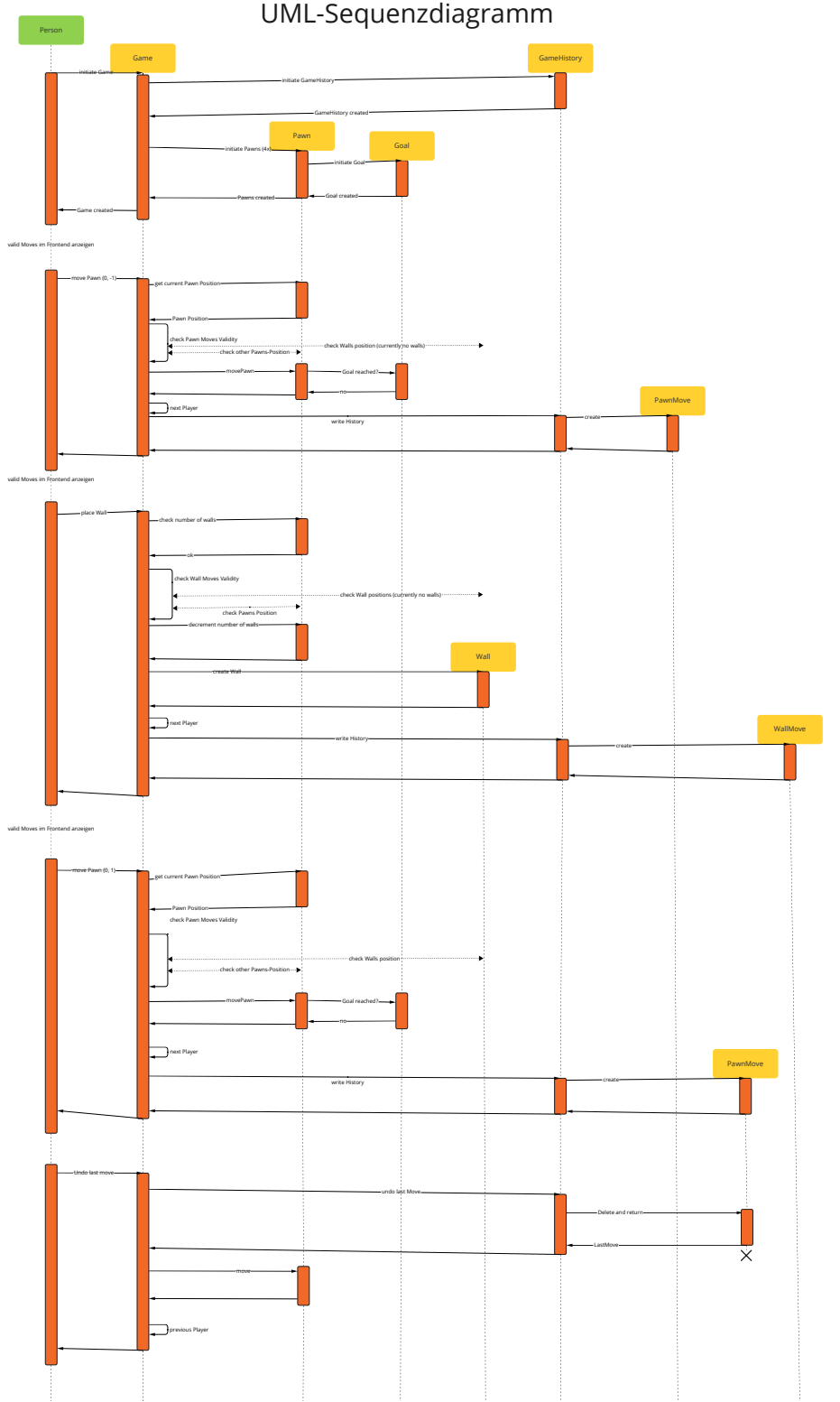
High-Level Architektur (3 Schichten / MVC)



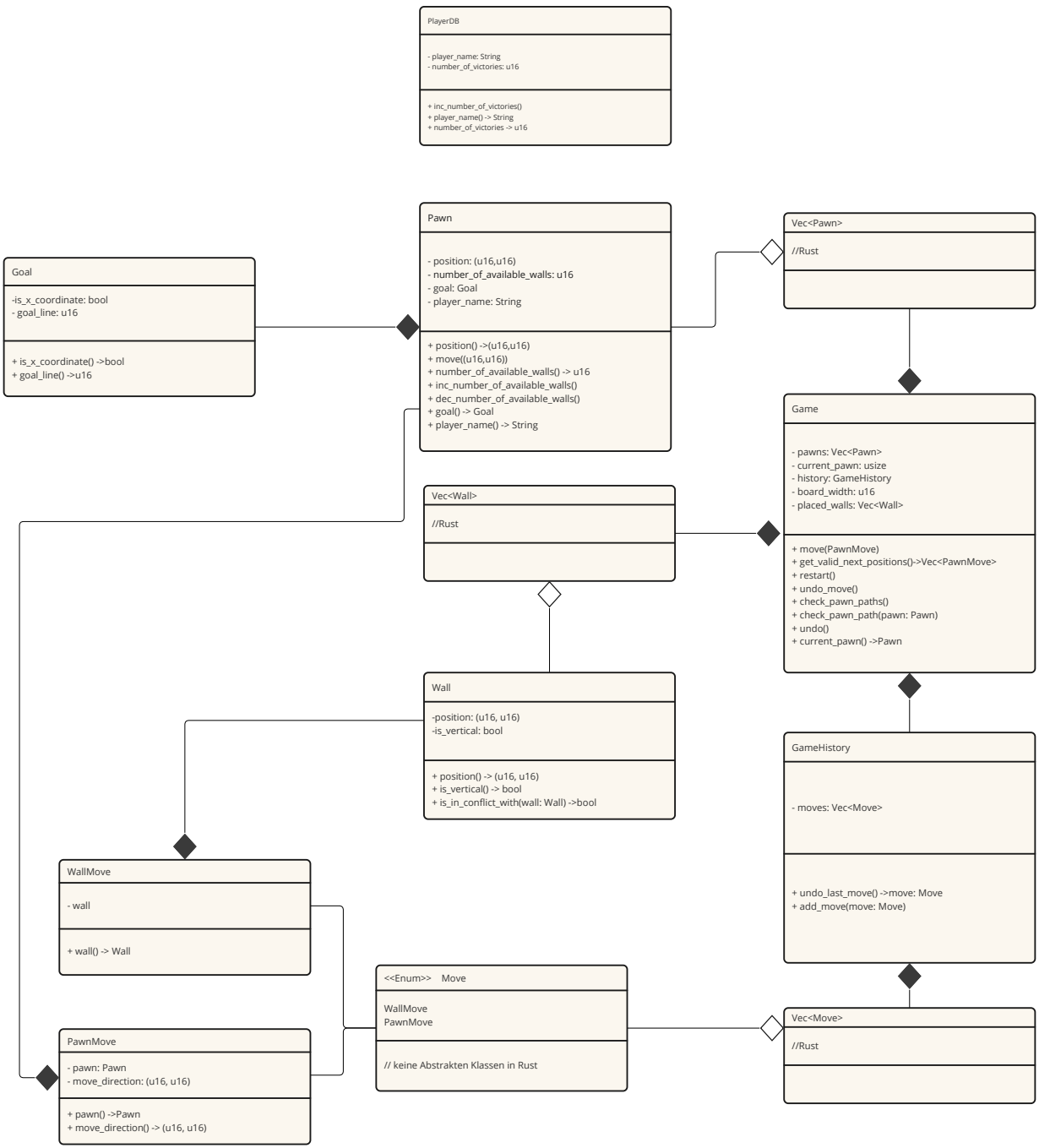
Zustandsdiagramm Spiel



UML-Sequenzdiagramm



UML-Klassendiagramm



Gib die Benutzer-Namen an

Spieler 1

Spieler 2

Spieler 3

Spieler 4

Start

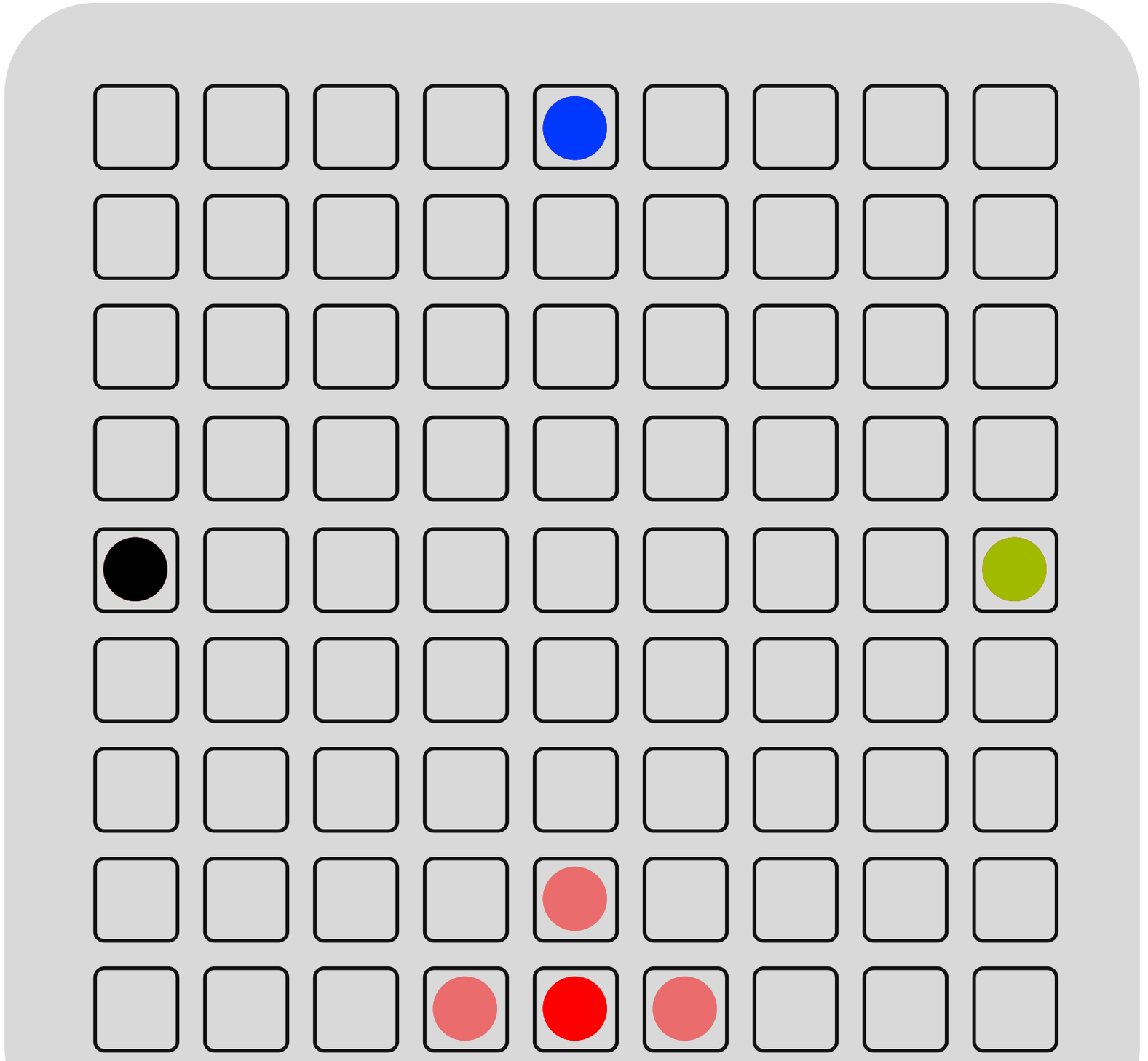
Spielregeln

Spieler 3

Wände: 10

Spieler 2

Wände: 10



Spieler 4

Wände: 10

Zurück

Spiel abbrechen

Spieler 1

Wände: 10

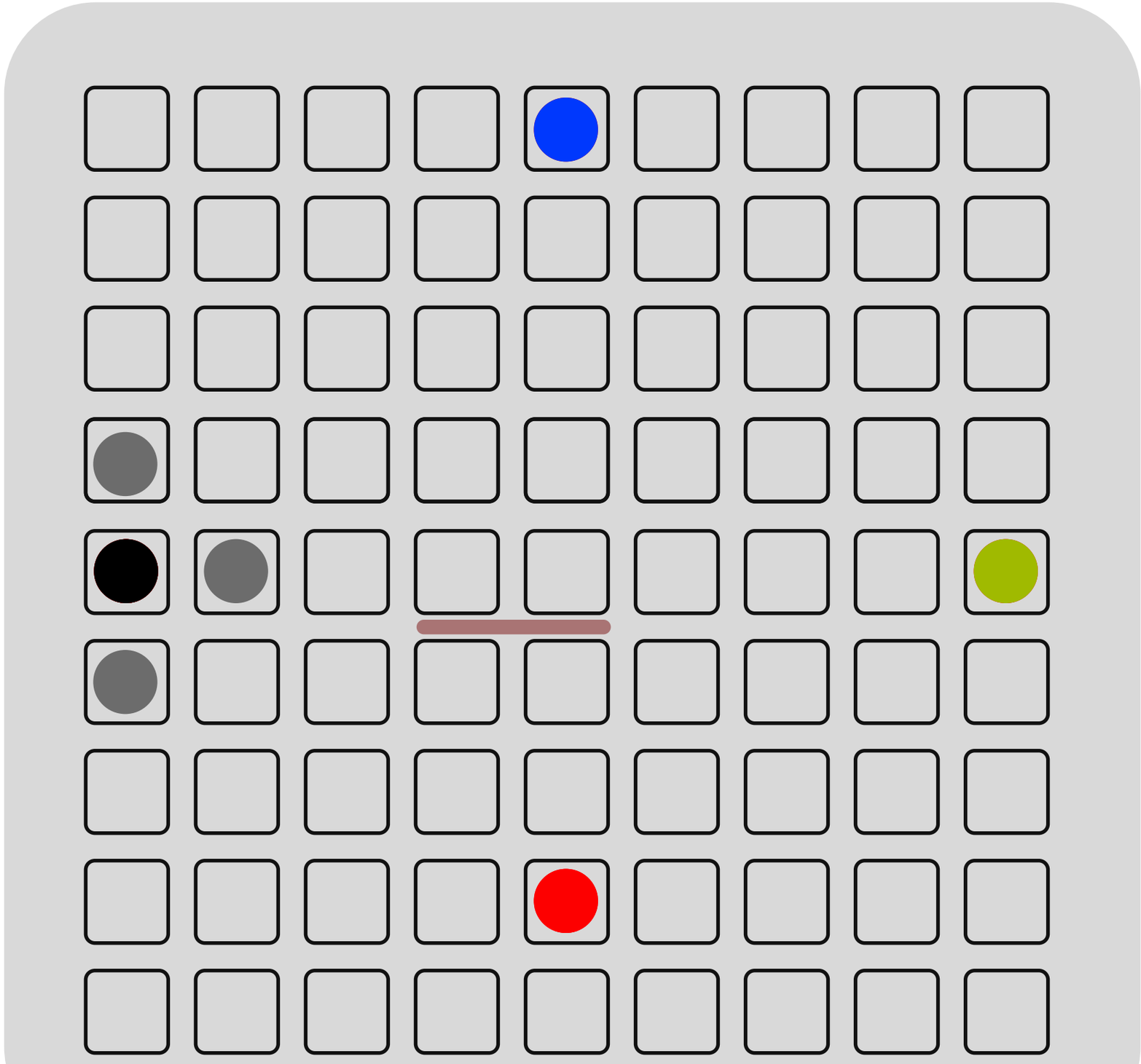
Spielregeln

Spieler 3

Wände: 10

Spieler 2

Wände: 10



Spieler 4

Wände: 10

Zurück

Spiel abbrechen

Spieler 1

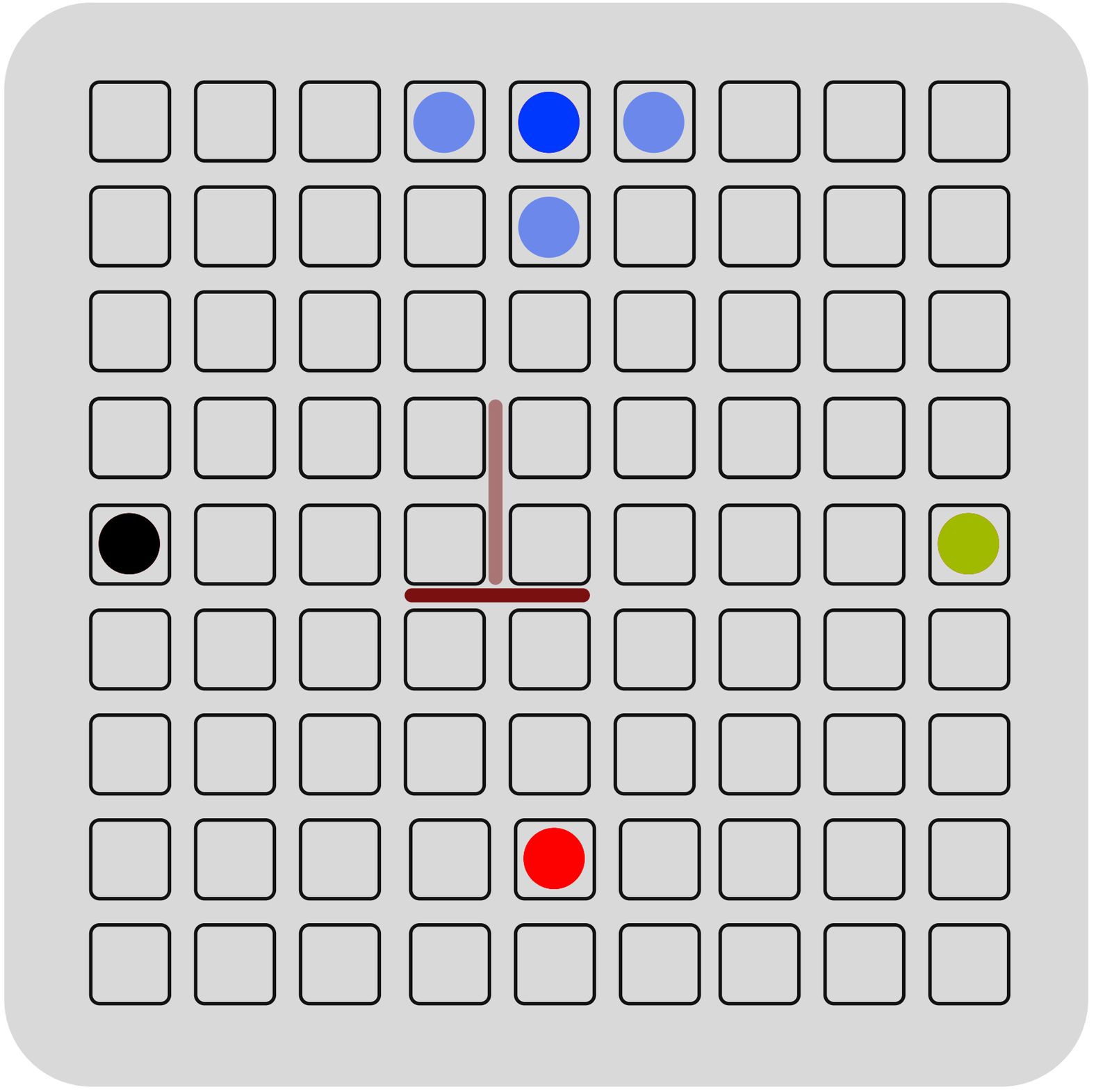
Wände: 10

Spielregeln

Spieler 3 Wände: 10

Spieler 2

Wände: 9



Spieler 4

Wände: 10

Zurück

Spiel abbrechen

Spieler 1 Wände: 10

Spielregeln

Spielregeln

Quoridor ist ein Brettspiel, dass mit zwei oder vier Personen gespielt werden kann. Klassischerweise ist das Spielbrett ein 9×9-Feld. Jeder Spieler startet mit seiner Spielfigur auf dem Mittelfeld einer Randspalte. Zusätzlich erhält jeder Spieler 10 Wände der länge 2 Spielfelder (bei 4 Spielern 6 Wände). Das Ziel ist es, die gegenüberliegende Randseite zu erreichen. Dafür ziehen die Spieler abwechselnd, beziehungsweise im Uhzeigersinn. Ein Zug beinhält entweder eine Bewegung der eigenen Spielfigur um eins in einer geraden Spielfeldrichtung (nicht schräg) oder das Setzen einer Wand zwischen 4 Felder. Eine Wand kann nicht durch eine Spielfigur passiert werden. Eine Wand kann nicht eine andere Wand schneiden. Auch dürfen Wände nicht so gesetzt werden, dass einer Spielfigur es nicht mehr möglich ist, die andere Seite zu erreichen. Es gewinnt, wessen Spielfigur als erstes die andere Seite erreicht.

Zurück

Zuordnung-Arbeitsaufteilung

- Auswahl Vorgehensmodell → Jared & Valentin
- Personas → Jared & Valentin
- User-Story-Map → Valentin, Jared: Kontrolle, Ergänzung
- Pflichtenheft → Jared, Valentin: Kontrolle, Ergänzung
- ER-Diagramm → Jared & Valentin
- Klassendiagramm → Jared & Valentin
- High-Level-Architektur → Jared & Valentin
- Sequenzdiagramm → Valentin, Jared: Kontrolle, Ergänzung
- Zustandsdiagramm → Valentin, Jared: Kontrolle, Ergänzung
- Entwurfsmuster → Jared, Valentin: Kontrolle, Ergänzung