

Projekt: Invers

Einführung Künstliche Intelligenz

Xander Van der Weken
FH-Erfurt

01. August 2023

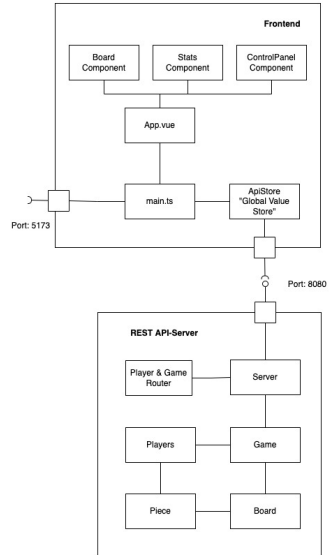
- ① Idee
- ② Aufbau
- ③ Frontend
- ④ Backend
 - Allgemein
 - Spiellogik
 - ComPlayer
- ⑤ Showcase

- Realisierung des Kombinatorischen Schiebepiels „Invers“
- Mensch gegen KI
- KI basierend auf einem Min-Max-Algorithmus mit Tiefe 3



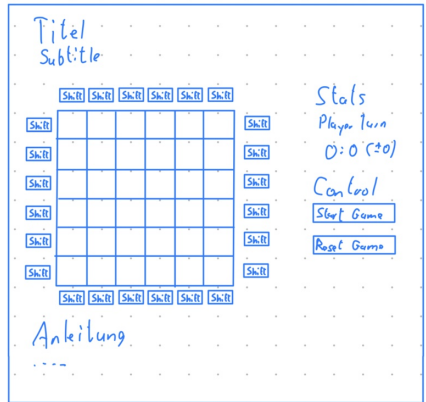
Systemarchitektur:

- System besteht aus 2 Komponenten
- *Frontend:*
 - VueJs3 Client mit Typescript
 - Dynamische Website zum Anzeigen des Spiels
- *Backend:*
 - Ktor Rest Server mit Kotlin
 - Server mit Spiellogik und Zusatzinformationen
 - Interagierbar über REST Schnittstelle



Anforderungen an das UI:

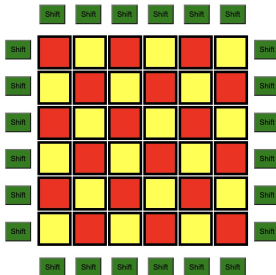
- 6x6 Spielfeld anzeigen
- 4 verschiedene Felder
- System zum Zug ausführen
- Legale und Illegale Züge anzeigen
- Spielstatus anzeigen
- Spielkontrolle



Resultat:

Inversi Spiel

Kombinatorisches Spiel, für das Modul "Einführung Künstliche Intelligenz", durch Xander Van der Weken



Statistics

Player 1 turn

18 : 18 (0)

Player 1 has 1 Pieces in Hand

Player 2 has 1 Pieces in Hand

Control Panel

Start Game

Reset Game

Options

Player 1 (red) is User Player

Player 2 (yellow) is AI Player

How To Play

Inversi ist ein abstraktes Spiel für zwei Spieler, bei dem die Spieler abwechselnd Spielsteine aus einem 6x6-Gitter schieben und umdrehen. Eine Seite ist einfarbig rot oder gelb, die andere ist ein roter oder gelber Punkt.

Beim nächsten Zug schieben Sie den umgekehrten Spielstein in das Gitter und zwingen einen anderen Spielstein heraus.

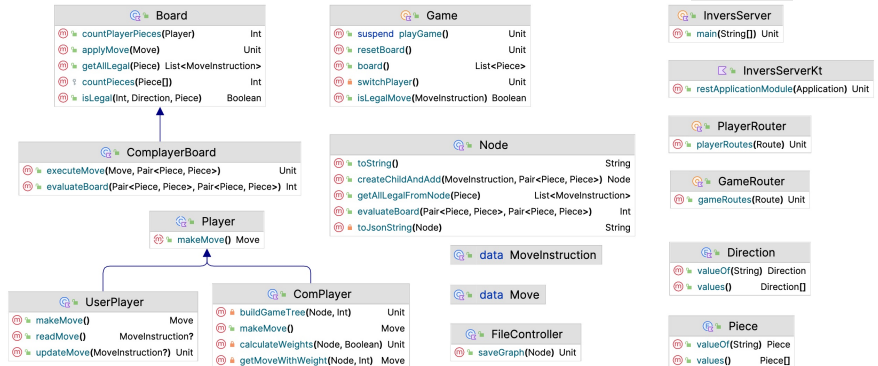
Man dreht die Spielsteine nur einmal um, um die Punktseite zu zeigen, und man kann den Punkt des Gegners nicht herausrutschen lassen.

Nach und nach verriegelt sich das Brett, es sei denn, Sie beginnen, Ihre eigenen umgedrehten Spielsteine herauszuschieben.

Der Gewinner ist der erste, der alle seine Steine umdreht, oder derjenige, der die meisten umgedreht hat, wenn keine weiteren Züge mehr möglich sind.

[Mehr Informationen](#)

Klassen UML Diagramm:



- Server besitzt die Spiellogik und REST-Schnittstelle
- REST-Schnittstelle bietet:
 - Nutzereingabe über die „/players“-Route
 - Spiel Ein- und Ausgaben über die „/game“-Route
- *Piece* und *Direction* sind Enums

MVC-Modell:

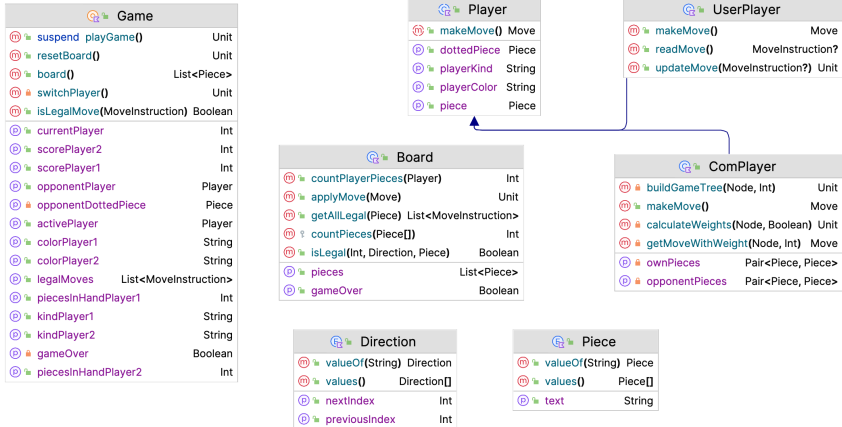
- Models: Player (UserPlayer, ComPlayer), Piece
- View: Board
- Controller: Game

playGame-Methode:

- repräsentiert den Game Loop
- Überprüft ob das Spiel vorbei ist
- Wenn der aktive Spieler einen Stein schieben kann, wird die *makeMove*-Methode des aktiven Spielers aufgerufen
- wechselt den Spieler

Game		
Ⓜ	suspend playGame()	Unit
Ⓜ	resetBoard()	Unit
Ⓜ	board()	List<Piece>
Ⓜ	switchPlayer()	Unit
Ⓜ	isLegalMove(MoveInstruction)	Boolean
Ⓟ	currentPlayer	Int
Ⓟ	scorePlayer2	Int
Ⓟ	scorePlayer1	Int
Ⓟ	opponentPlayer	Player
Ⓟ	opponentDottedPiece	Piece
Ⓟ	activePlayer	Player
Ⓟ	colorPlayer1	String
Ⓟ	colorPlayer2	String
Ⓟ	legalMoves	List<MoveInstruction>
Ⓟ	piecesInHandPlayer1	Int
Ⓟ	kindPlayer1	String
Ⓟ	kindPlayer2	String
Ⓟ	gameOver	Boolean
Ⓟ	piecesInHandPlayer2	Int

Klassen UML Diagramm:

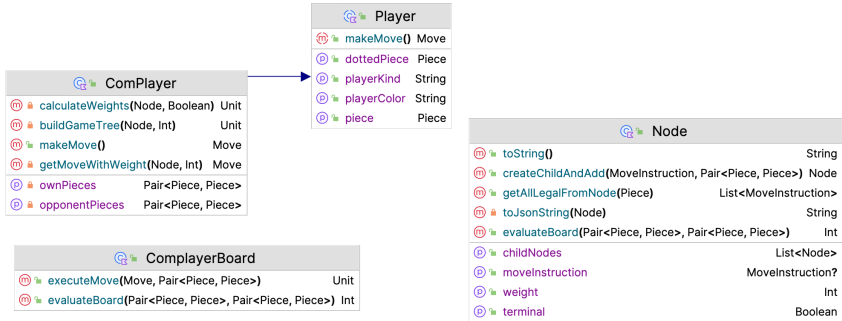


Vorgehensweise:




Die *MakeMove*-Methode der KI ruft folgende Hilfsmethoden auf:

- ① Baum aufbauen (*buildGameTree*-Methode)
 - Rekursiv Baum der Tiefe 3 aufbauen
 - Baum besteht aus *Node*'s
- ② Baum evaluieren (*calculateWeights*-Methode)
 - Rekursive Funktion
 - Bei Terminal Node wird das „aktuelle“ Board evaluiert
 - Sonst wird von den Kindern die Gewichte Maxi- oder Minimiert
- ③ Besten Move suchen (*getMoveWithWeight*-Methode)
 - Sucht zu dem besten Gewicht eine passende *MoveInstruction* und gibt die zurück

Klassen UML Diagramm:



Docker starten, und eine Runde spielen

▼	 eki_invers_vue		Running (2/2)	0.22%		12 seconds ago	■	:	🗑
	 Invers_UI 94eaa687777f1 🗑	eki_invers_vue-client	Running	0%	8081:80 🔗	12 seconds ago	■	:	🗑
	 Invers_Server edcfa7db3f08 🗑	eki_invers_vue-server	Running	0.22%	8080:8080 🔗	13 seconds ago	■	:	🗑

**Vielen Dank für Ihre
Aufmerksamkeit
Noch Fragen?**