Hochschule für Technik
und Wirtschaft Berlin

University of Applied Sciences

MBSNW - Technical White Paper - Bluetooth Schach - Version 4

WiSe 2023/24

Hochschule für Technik und Wirtschaft (HTW) Berlin
Fachbereich 4: Informatik, Kommunikation und Wirtschaft
Studiengang *Angewandte Informatik*

Dozent_in: Thomas Schwotzer

Eingereicht von

Dennis Forster [Matr. Nr. 586888]

07.01.2025

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Projektbeschreibung

Ziel des Projekts ist es, im Rahmen der Veranstaltung „Mobile Betriebssysteme und Netzwerke“ an der HTW Berlin im WiSe 2024/25 eine semesterbegleitende Leistung im Rahmen einer Android Anwendung zu erbringen. Dafür soll eine Schach-Applikation geschrieben werden, dessen Datenaustausch auf Bluetooth basiert.
Das Repository dazu befindet sich unter: https://github.com/Semanticraft/bluetooth-chess.

1.1 Nutzen

Mit einer auf Bluetooth basierten Datenübertragung lässt sich mit Freunden und Familie Schach spielen, auch dann, wenn keine Internet-Verbindung vorhanden ist. Man denke zum Beispiel an den Flugmodus in einem Flugzeug oder an einem Aufenthalt in ländlichen Regionen.

1.2 Einhaltung der Anforderungen des Auftraggebers

- Die genutzte Programmiersprache wird, gemäß des Standards im Studiengang Angewandte Informatik an der HTW Berlin, Java sein.
- Die GUI wird nicht trivial sein, da im Rahmen der Anzeige der möglichen Bluetooth-Verbindungen eine Recycler View eingesetzt wird.
- Die App wird den Spielstand nach jedem Halbzug[1] persistent lokal speichern, damit dieser bspw. bei Abbruch der Verbindung wiederhergestellt werden können.
- Die Komponentenaufteilung wird MVC folgen.
- Die App wird getestet durch Unit-, GUI- und Integrationstests.
- Die App nutzt Bluetooth basierte Datenübertragung und vertieft sich somit auf Ad-Hoc Netzwerke.

2 Anwendungsfalldiagramme

Die Anwendungsfälle werden für jede View in je einem Use Case Diagramm visualisiert, für eine bessere Anschaulichkeit.

[1] Ein Halbzug ist der Zug, den *ein* Spieler macht. Eine Runde ist ein vollständiger Zug.

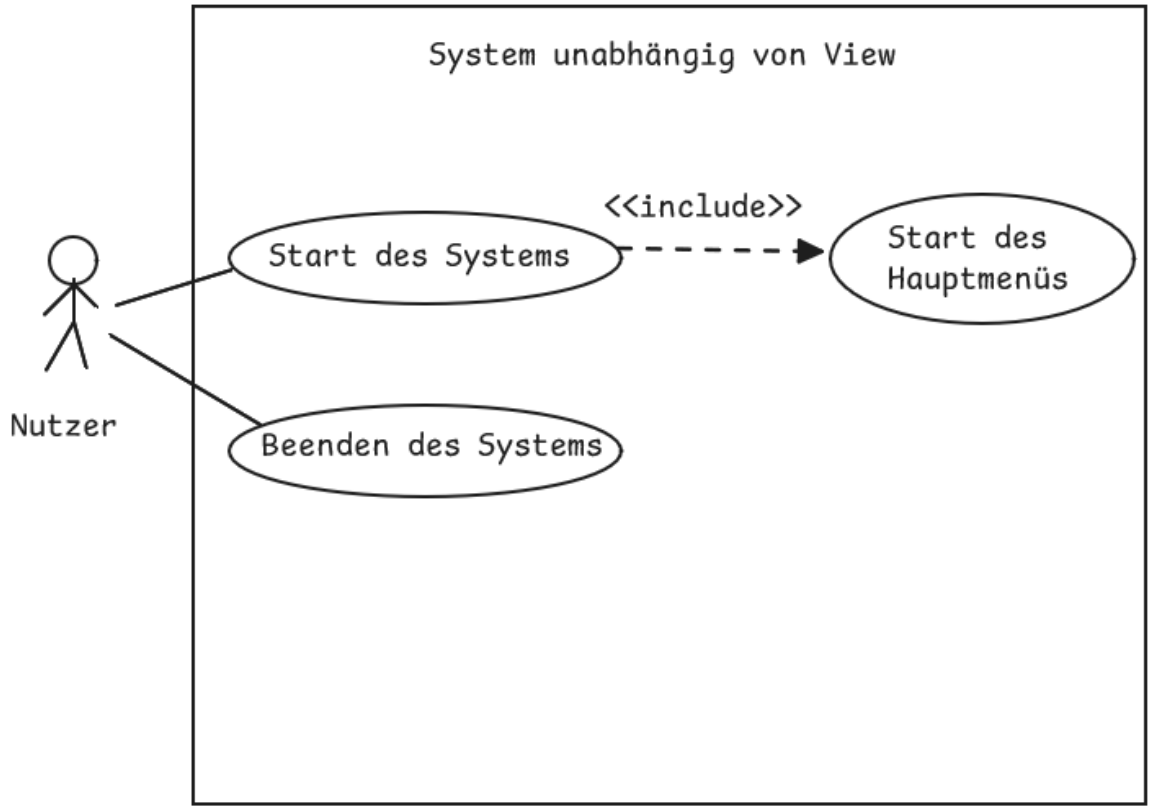

Abbildung 1: Use Case Diagramm für alle Anwendungsfälle, die unabhängig zur View sind

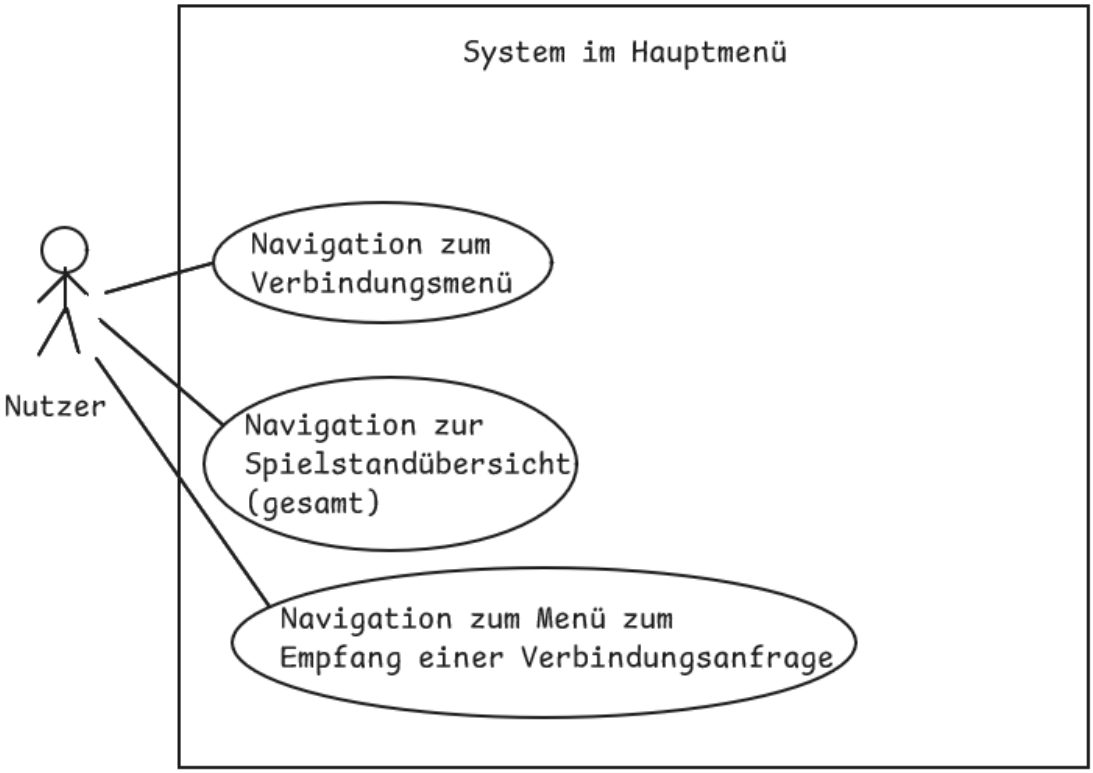

Abbildung 2: Use Case Diagramm für alle Anwendungsfälle im Hauptmenü

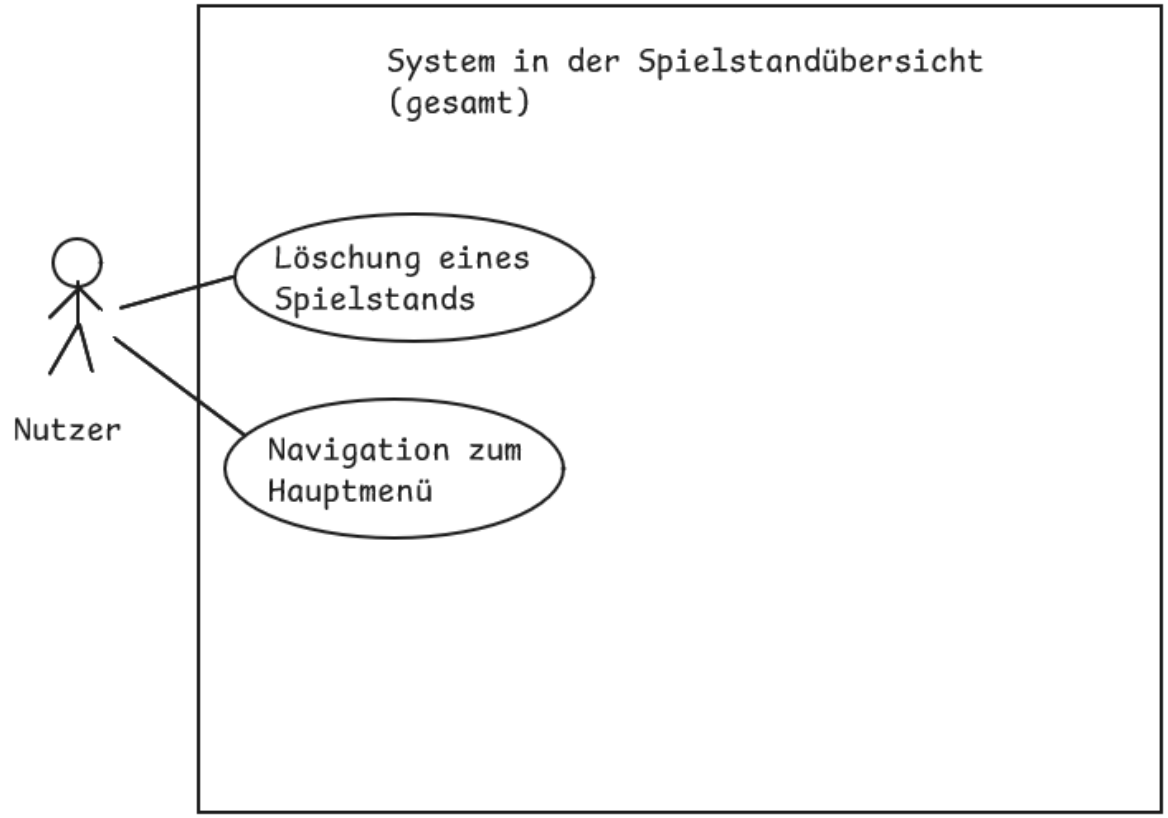

Abbildung 3: Use Case Diagramm für alle Anwendungsfälle zur Spielstandübersicht (gesamt)

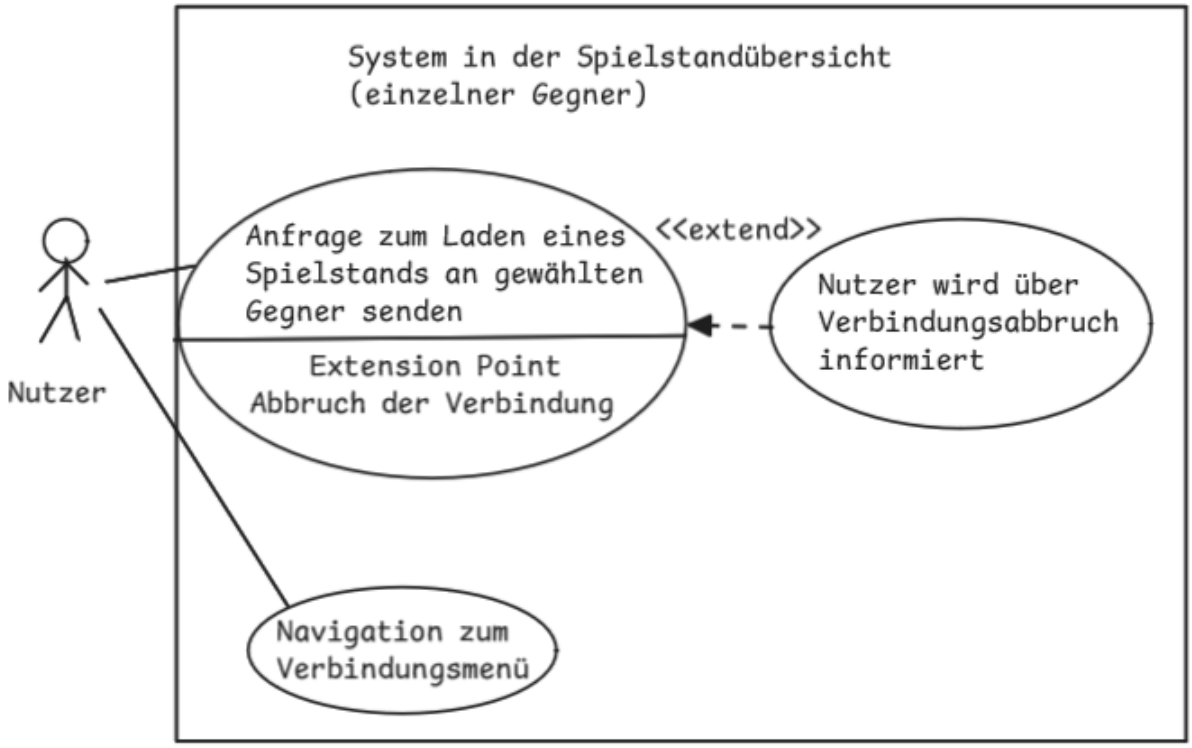

Abbildung 4: Use Case Diagramm für alle Anwendungsfälle zur Spielstandübersicht (einzelner Gegner)

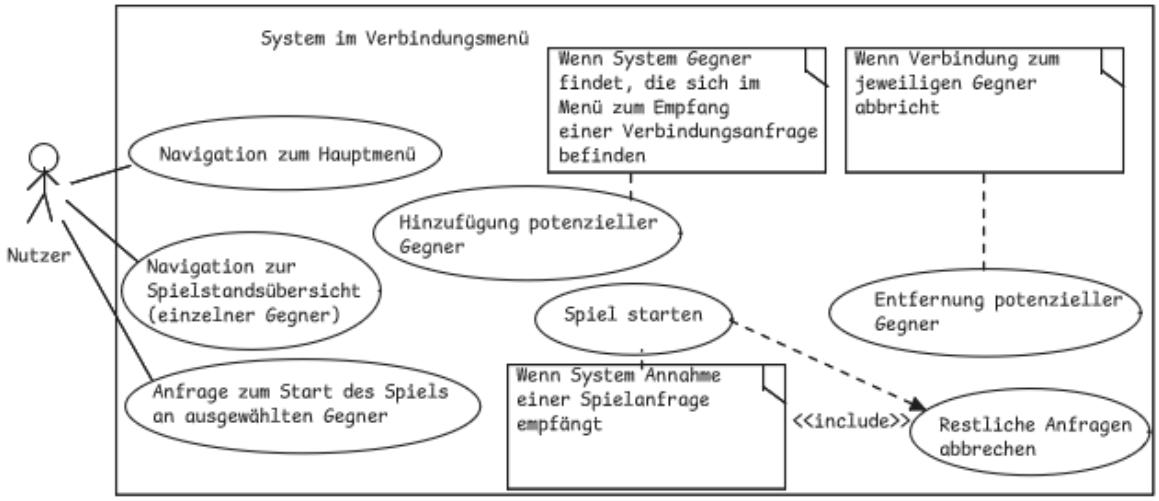

Abbildung 5: Use Case Diagramm für alle Anwendungsfälle im Verbindungsmenü

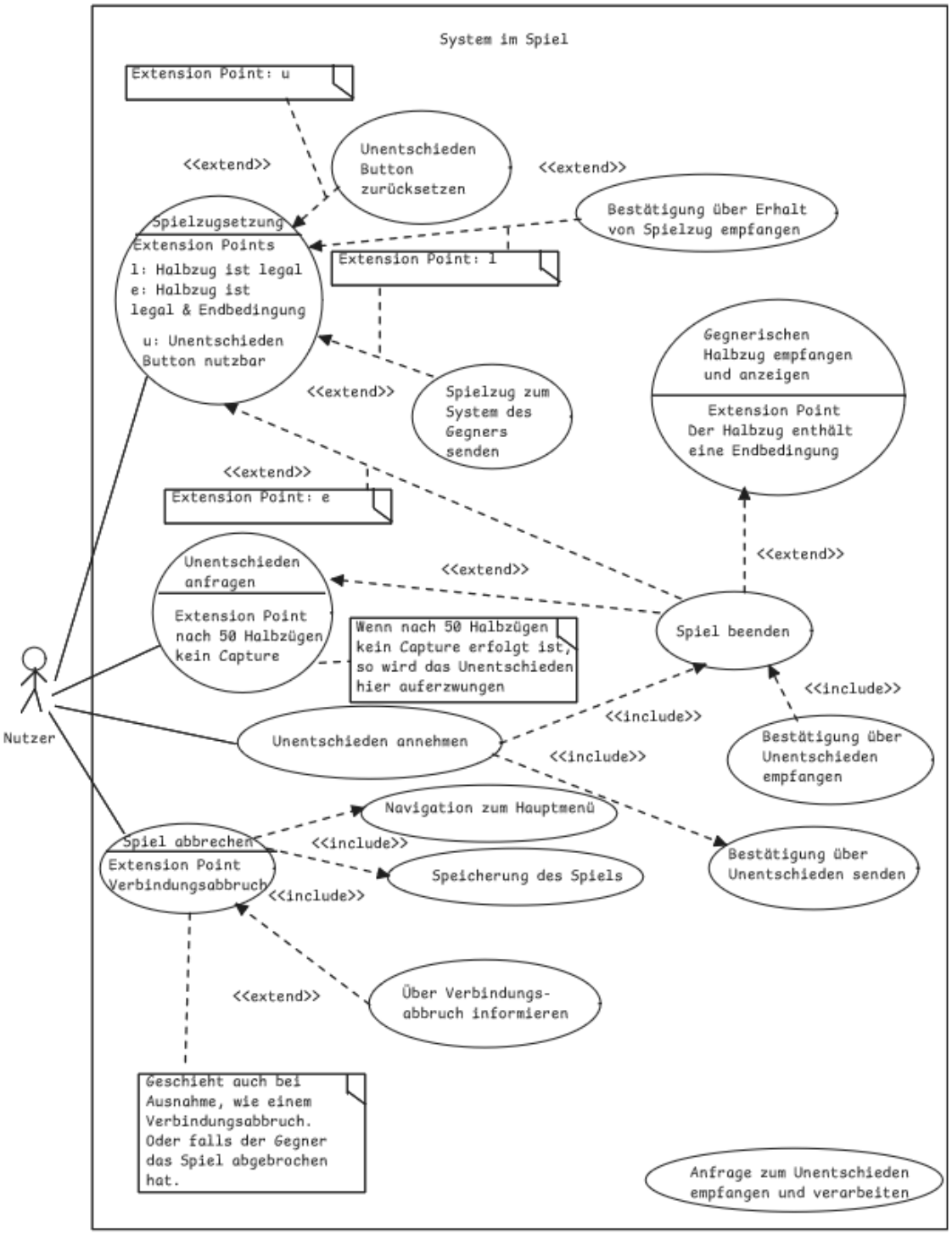

Abbildung 6: Use Case Diagramm für alle Anwendungsfälle im Spiel

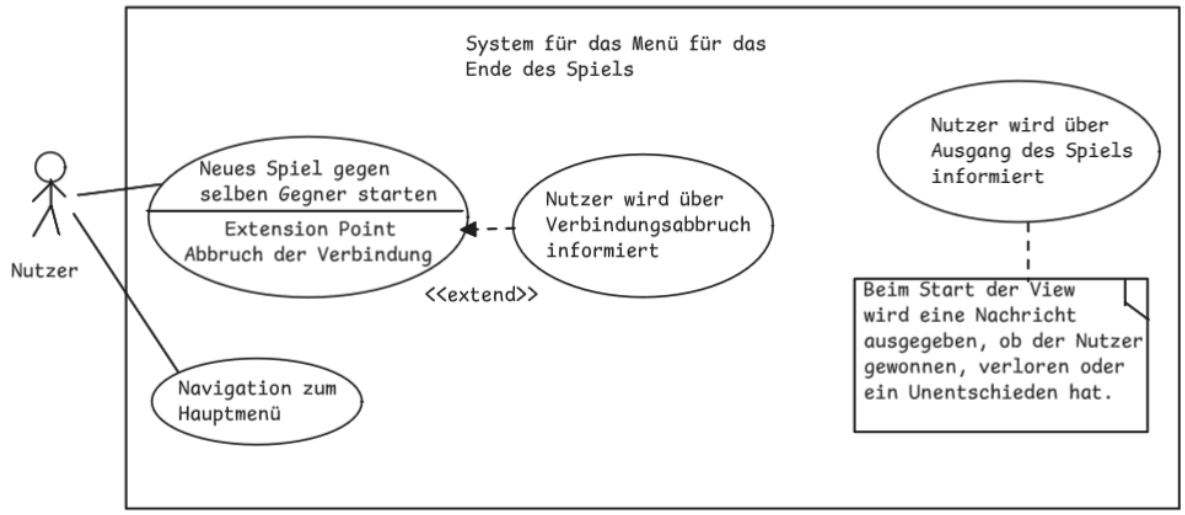

Abbildung 7: Use Case Diagramm für alle Anwendungsfälle im Menü für das Ende des Spiels

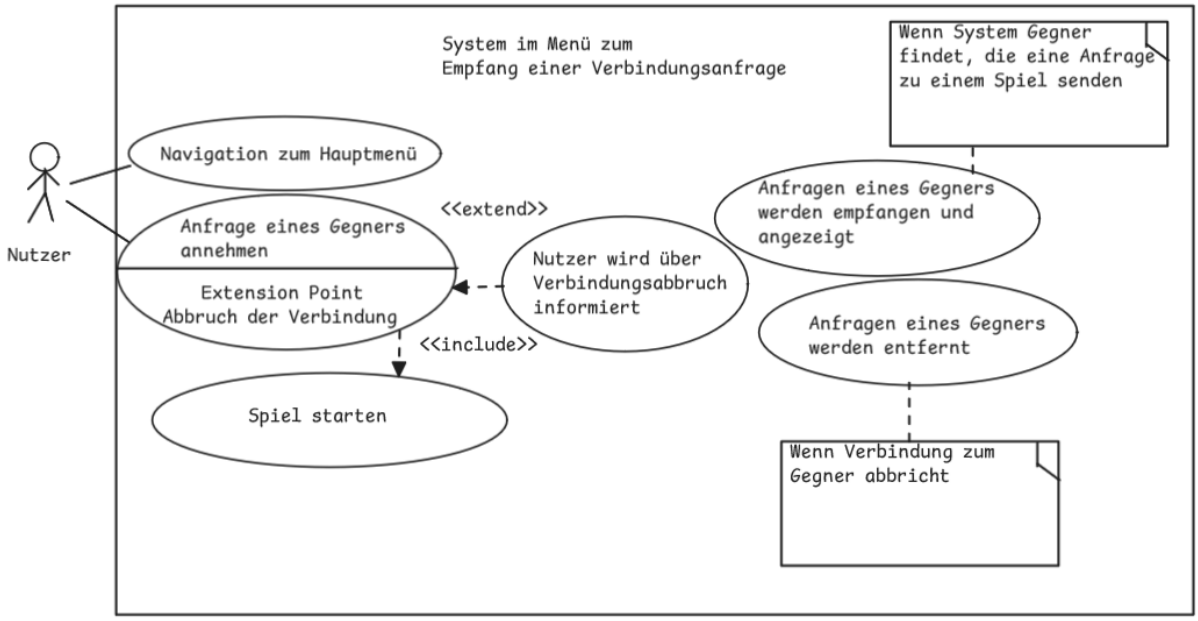

Abbildung 8: Use Case Diagramm für alle Anwendungsfälle im Menü zum Empfang einer Verbindungsanfrage

3 Anwendungsfälle die (noch) nicht Teil des Systems werden sollen

Vor jedem Anwendungsfall symbolisiert dessen Zahl, in welchem der Views die Anforderung durchgeführt werden soll. 1 bedeutet Hauptmenü, 2 Spielstandübersicht (gesamt), 3 Spielstandübersicht (einzelner Gegner), 4 Verbindungsmenü, 5 Spiel, 6 Menü für das Ende des Spiels, 7 Menü zum Empfang einer Verbindungsanfrage, X für Unabhängigkeit von View.

- 5: Ein Timer wird für jeden Spieler angezeigt und vom System in Abhängigkeit zum Spieler, der gerade einen Zug ausführen soll, jede Sekunde aktualisiert
- 5: Wenn der Timer eines Spielers abläuft, verliert dieser
- 5: Unentschieden durch Threefold Repetition wird vom System erkannt
- 5: Unentschieden durch Dead Position wird auch außerhalb von Insufficient Material erkannt
- 5: Nutzer führt En Passant aus
- 5: Nutzer führt Castling aus
- 5: Nutzer führt Promotion aus
- 5: Nutzer kann auswählen, ob dieser das Spiel speichern möchte oder nicht, bei absichtlichem Abbruch des Spiels
- 5: Zughistorie wird angezeigt und nach jedem Halbzug vom System aktualisiert
- 5: Bei Verbindungsabbruch im Spiel kann ein Spieler selbst entscheiden, ob dieser zum Hauptmenü zurücknavigieren oder warten will, dass sich die Verbindung wieder aufbaut.
- 5: Nutzer kann das Spiel aufgeben.
- 1: Navigation zu und von Spieloptionen und Anpassung dieser durch den Nutzer

4 GUI Mock-Ups

Hier folgen Skizzen der Views und der Navigationsmöglichkeiten, basierend auf den funktionalen Anforderungen:

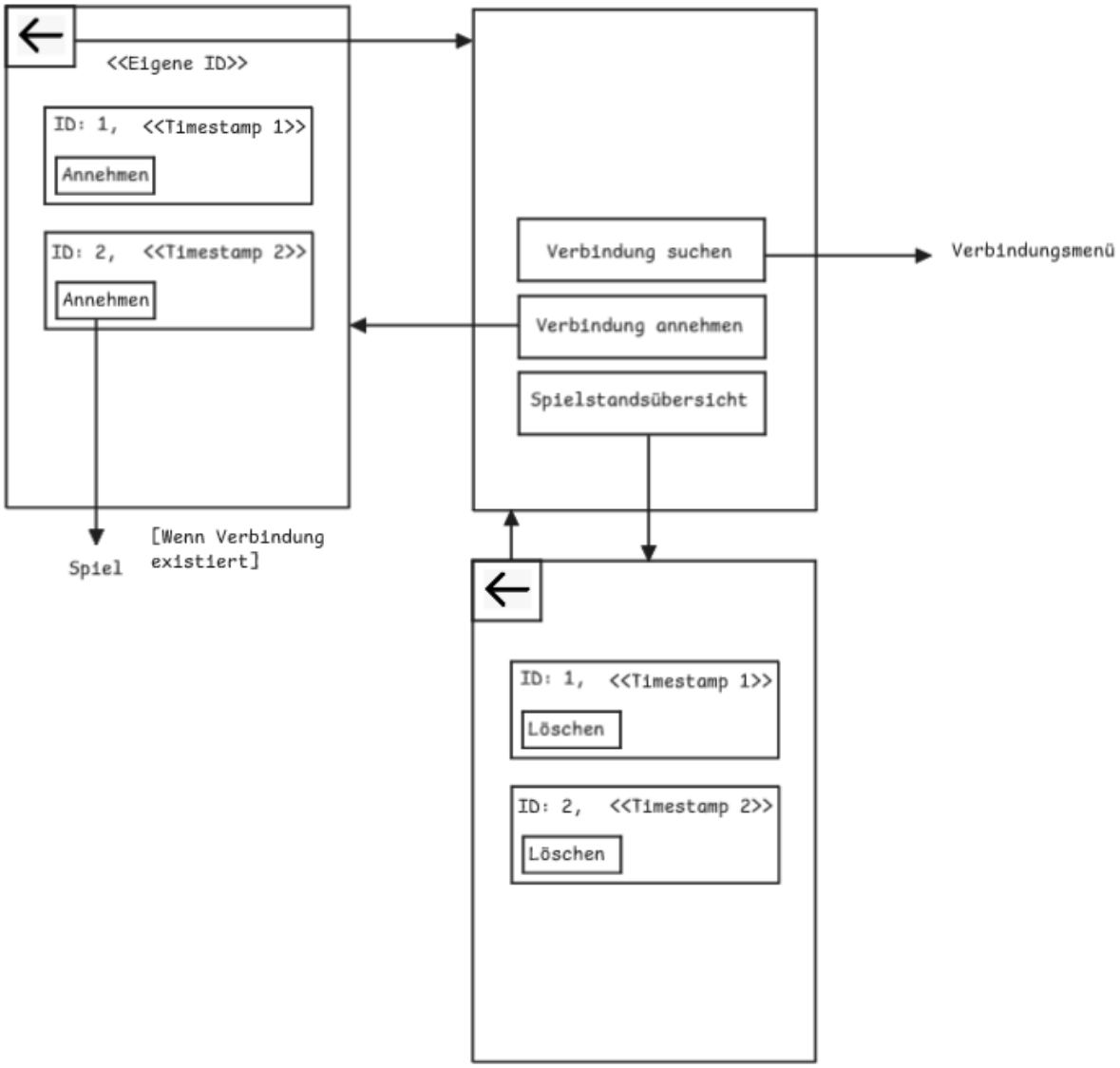

Abbildung 9: UI Mock-Up 1

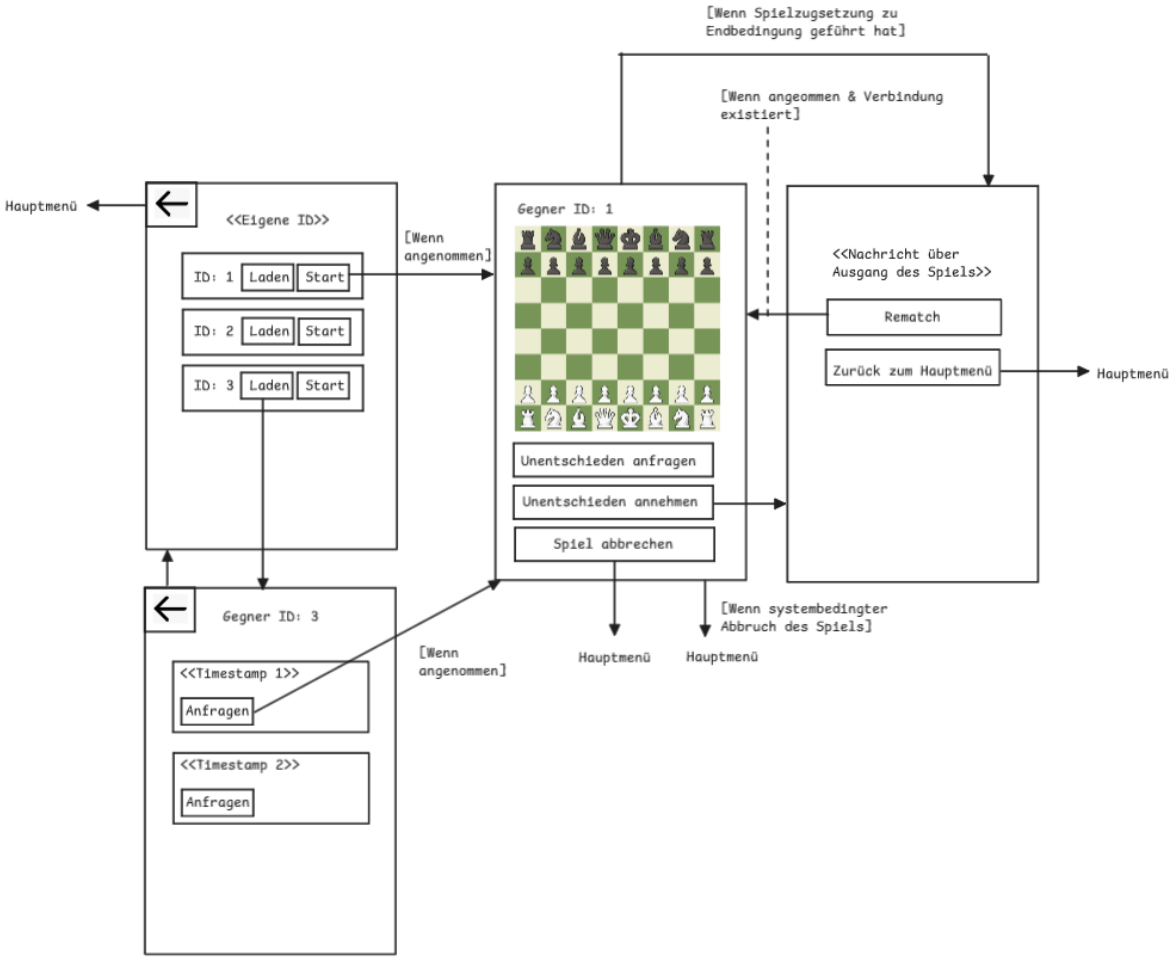

Abbildung 10: UI Mock-Up 2, Bild des Schachbretts entnommen aus [1]

5 GUI-Testkonzept

Mithilfe des Espresso Frameworks lassen sich die Views einfach automatisiert testen. Ein grobes UI Testkonzept der unterschiedlichen Views soll in diesem Abschnitt dargelegt werden. Die Korrektheit wird hierbei im Bezug zur Einhaltung der Mock-Ups der GUI und der funktionalen Anforderungen gemessen. Die Tests können im Code weiter aufgespalten werden.

View	Test	Bestanden (j), nicht bestanden (n), noch nicht geschrieben (nng)
Hauptmenü	Die Buttons enthalten die korrekten Texte	j (testButtonText)
Hauptmenü	Die Navigation der Buttons funktioniert korrekt	j (testButtonNavigationSSH, testButtonNavigationCA, testButtonNavigationC)
Menü zum Empfang einer Verbindungsanfrage	Es werden die korrekten Gegner IDs und Timestamps angezeigt	nng
Menü zum Empfang einer Verbindungsanfrage	Die Navigation der Buttons funktioniert korrekt	nng
Menü zum Empfang einer Verbindungsanfrage	Scrolling ist möglich, wenn die Anfragen über den unteren Bildschirmrand hinaus gehen	nng
Spielstandsübersicht (gesamt)	Es werden die korrekten Gegner IDs und Timestamps angezeigt	nng
Spielstandsübersicht (gesamt)	Die Navigation der Buttons funktioniert korrekt	nng
Spielstandsübersicht (gesamt)	Scrolling ist möglich, wenn die Spielstände über den unteren Bildschirmrand hinaus gehen	nng
Spielstandsübersicht (gesamt)	Die Löschung von Spielständen funktioniert korrekt	nng
Verbindungsmenü	Die Navigation der Buttons funktioniert korrekt	nng
Verbindungsmenü	Es werden die korrekten Gegner IDs angezeigt	nng
Spielstandsübersicht (einzelner Gegner)	Die Navigation der Buttons funktioniert korrekt	nng
Spielstandsübersicht (einzelner Gegner)	Es werden die korrekten Timestamps und die korrekte Gegner ID angezeigt	nng

Tabelle 1: UI Testkonzept für die einzelnen Views von Hauptmenü bis Spielstandsübersicht (einzelner Gegner)

View	Test	Bestanden (j), nicht bestanden (n), noch nicht geschrieben (nng)
Spiel	Die Navigation der Buttons und des Systems funktioniert korrekt	nng
Spiel	Es wird die korrekte Gegner ID angezeigt	nng
Spiel	Die Spielzugsetzung funktioniert korrekt	nng
Menü für das Ende des Spiels	Die Navigation der Buttons funktioniert korrekt	nng
Menü für das Ende des Spiels	Es wird der korrekte Spielausgang angezeigt	nng

Tabelle 2: UI Testkonzept für die einzelnen Views von Spiel bis Menü für das Ende des Spiels

Aufgrund von Zeitmangel wurde der Großteil letztlich manuell überprüft und bestätigt. Das einzige Problem besteht noch bei der korrekten Spielzugsetzung (aufgrund der evaluate Methode von GameRules), dem nicht korrekten Spielausgang (ebenfalls aufgrund von evaluate und weil noch bei beiden Spielern immer dieselbe und keine unterschiedliche Nachricht angezeigt wird) und bei den noch nicht funktionierenden Draw Requests. Außerdem wirkt der Broadcast Receiver noch nicht korrekt, wodurch ein Spieler im Spiel bleibt, wenn der andere es abbricht, bis dieser das Spiel selbst abbricht oder versucht, einen Zug auszuführen.

6 Komponentendiagramm

Aus den vorherigen Phasen ergibt sich das folgende Komponentendiagramm (Die BluetoothEngine heißt jetzt Connection Engine):

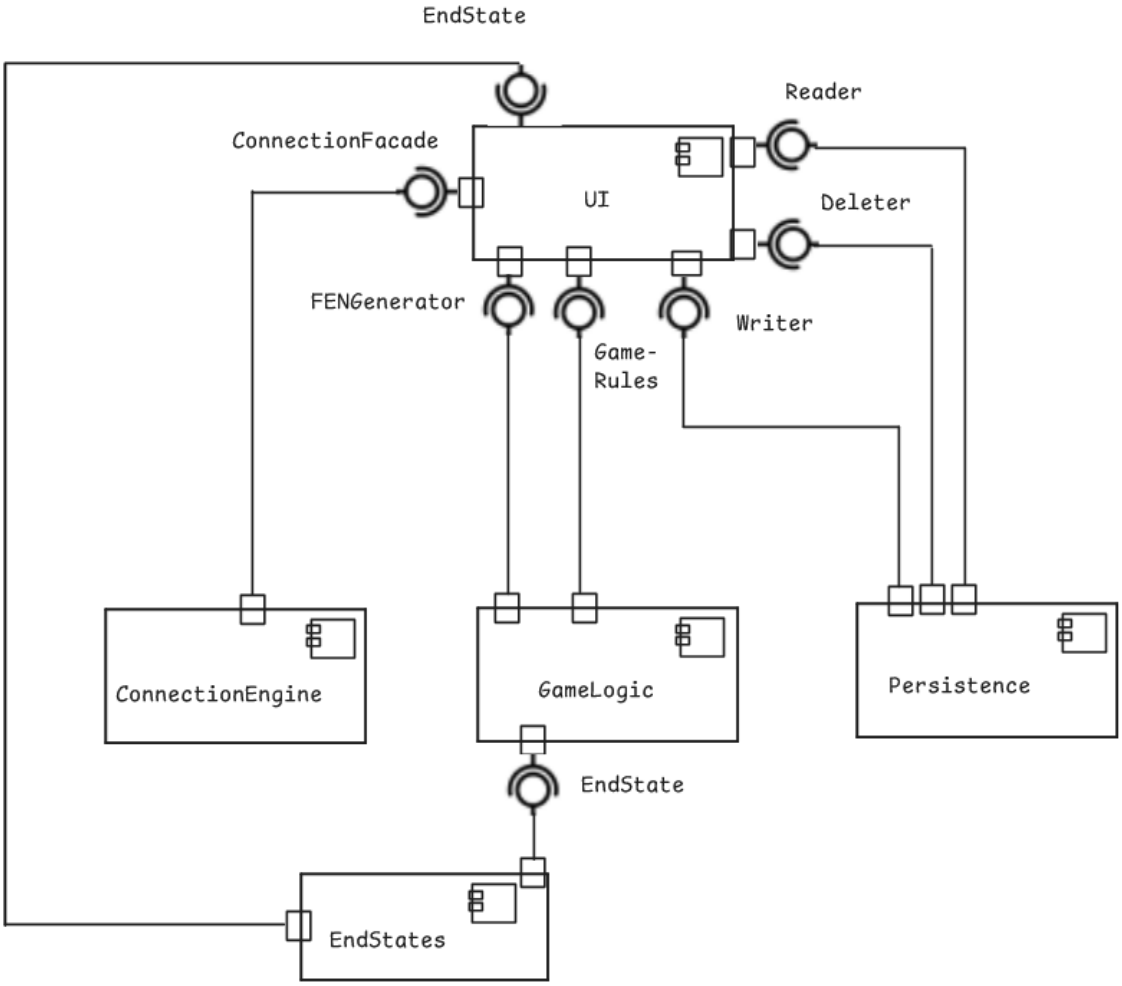

Abbildung 11: Komponentendiagramm

Die UI unterteilt sich dabei in Subkomponenten mit der jeweiligen View und den dazugehörigen Controllern und Models und ist die zentrale Komponente des Systems. Die Komponenten sind insgesamt Schichtweise im Bezug auf Dependencies angeordnet und können somit Bottom-Up entwickelt und mit Testtreibern integriert werden, wodurch aufwendiges Mocking in der Regel nicht notwendig wird. Ein Navigationsmanager ist nicht notwendig, da Android bereits über Navigationsmethoden verfügt.
In den nächsten Abschnitten wird näher auf die einzelnen Komponenten und deren Schnittstellen eingegangen.

7 GameLogic

Die GameLogic Komponente soll dem Controller des Spiels dienen, um festzustellen, ob ein Halbzug legal ist oder nicht und ob ein Halbzug zu einer Endbedingung führt. Auch soll ein FEN String Generator dabei helfen, das Board zu initialisieren und zu speichern.

7.1 Schnittstellenmethoden

Schnittstelle	GameRules
Methode	evaluate
Eingabe	startRow: int, startCol: int, endRow: int, endCol: int, startState: int[][], endState: int[][]
Ausgabe	MoveResult
Beschreibung	Nimmt den Start- und Endzustand, sowie die Anfangs- und Endposition der bewegten Figur auf und überprüft, ob der Übergang legal ist und ob dieser zu einer Endbedingung des Spiels führt. Der Ausgang der Überprüfung wird in moveResult ausgegeben.
Ausnahmen	/

Tabelle 3: evaluate Methode der GameRules Schnittstelle

Schnittstelle	FENGenerator
Methode	initGame
Eingabe	color: String
Ausgabe	int[][]
Beschreibung	Nimmt eine Farbe „black“ oder „white“ und gibt basierend darauf den Startzustand des Spielbretts aus.
Ausnahmen	/

Tabelle 4: initGame Methode der FENGenerator Schnittstelle

Schnittstelle	FENGenerator
Methode	getFEN
Eingabe	state: int[][]
Ausgabe	String
Beschreibung	Nimmt eine Spielbrettkonfiguration „state“ und generiert basierend darauf einen FEN-String des Spielbretts. Dies ist hilfreich zur Speicherung des Spiels.
Ausnahmen	/

Tabelle 5: getFEN Methode der FENGenerator Schnittstelle

Schnittstelle	FENGenerator
Methode	getState
Eingabe	FEN: String
Ausgabe	int[][]
Beschreibung	Nimmt einen FEN-String auf und generiert basierend darauf eine Spielbrettkonfiguration „state“. Dies ist hilfreich zum Laden eines Spielstands aus einem File mit FEN-Strings.
Ausnahmen	/

Tabelle 6: getState Methode der FENGenerator Schnittstelle

8 Persistence

Die Persistence Komponente dient der Speicherung, der Löschung und dem lesen von Spielständen. Zu Spielständen gehört die Spielbrettkonfiguration, ein Timestamp und die ID des Gegners.

8.1 Schnittstellenmethoden

Schnittstelle	Reader
Methode	read
Eingabe	filename: String
Ausgabe	String
Beschreibung	Nimmt filename auf und liest den Inhalt des Files. Dieser Inhalt wird als String zurückgeliefert.
Ausnahmen	I/O-Exception.

Tabelle 7: read Methode der Reader Schnittstelle

Schnittstelle	Writer
Methode	saveGame
Eingabe	fileName: String, gameState: int[][], enemyID: long
Ausgabe	/
Beschreibung	Schreibt in das gegebene File den gameState, gefolgt von enemyID und timestamp als Zeile in eine CSV-Datei. Wenn das File nicht existiert, wird es angelegt.
Ausnahmen	I/O-Exception.

Tabelle 8: write Methode der Writer Schnittstelle

Schnittstelle	Writer
Methode	writeID
Eingabe	fileName: String
Ausgabe	/
Beschreibung	Schreibt in das gegebene File den momentanen Timestamp, falls das File leer ist, oder kreiert das File mit dem Timestamp als Eintrag. Dies ist hilfreich, um eine Identifikation der Spieler vorzunehmen.
Ausnahmen	I/O-Exception.

Tabelle 9: writeID Methode der Writer Schnittstelle

Schnittstelle	Writer
Methode	saveGame
Eingabe	fileName: String, gameState: String, enemyID: long
Ausgabe	/
Beschreibung	Schreibt in das gegebene File die gegnerische ID, einen aktuellen Timestamp und den gegebenen gameState in FEN. Falls das File leer ist, wird es mit dem Eintrag kreirt.
Ausnahmen	I/O-Exception.

Tabelle 10: saveGame Methode der Writer Schnittstelle

Schnittstelle	Deleter
Methode	delete
Eingabe	filename: String, timestamp: String
Ausgabe	/
Beschreibung	Löscht Eintrag mit dem jeweiligen Timestamp aus dem gegebenen File.
Ausnahmen	I/O-Exception.

Tabelle 11: delete Methode der Deleter Schnittstelle

9 Connection Engine (Alt: BluetoothEngine)

Die Connection Engine hilft bei der Einrichtung der Vorraussetzungen eines Bluetooth Verbindungsaufbaus, der Device Discovery, dem Verbindungsaufbau selbst, dem senden von PDUs über die Verbindung und dem Abbau der Verbindung. Sie gibt außerdem Auskunft über den Status der Verbindung über Broadcast Receiver.

9.1 Schnittstellenmethoden

Schnittstelle	ConnectionFacade
Methode	getInstance
Eingabe	playerID: long, context: Context
Ausgabe	ConnectionFacade
Beschreibung	Liefert Activity übergreifend eine Instanz einer ConnectionFacade und handhabt die Änderung der Activity durch das neu setzen des PropertyChangeListeners, das setzen des receivers für Verbindungsänderungen und das starten des AcceptThreads der den InputStream handhabt. Initialisiert auch den Bluetooth Adapter, falls dies noch nicht getan wurde.
Ausnahmen	/

Tabelle 12: getInstance Methode der ConnectionFacade Schnittstelle

Schnittstelle	ConnectionFacade
Methode	startEngine
Eingabe	discoverable: boolean
Ausgabe	/
Beschreibung	Setzt Vorraussetzungen für den Verbindungsaufbau. Wenn das Gerät unter der jeweiligen Activity discoverable sein soll, so werden die Voraussetzungen zur Sichtbarkeit für die Bluetooth Central geschaffen. Andernfalls wird das Gerät die Discovery starten und die devices Liste der Engine füllen und die Activity bezüglich gefundener Geräte benachrichtigen.
Ausnahmen	Bluetooth wird auf dem Gerät nicht unterstützt. Es läuft schon eine Bluetooth Operation. IOException, InterruptedException.

Tabelle 13: startEngine Methode der ConnectionFacade Schnittstelle

Schnittstelle	ConnectionFacade
Methode	stopEngine
Eingabe	context Context
Ausgabe	/
Beschreibung	Sorgt für einen ordnungsgemäßen Verbindungsabbau, für den Fall, dass zu einer anderen Activity navigiert wird, in der die Verbindung nicht mehr sinnvoll ist. Oder auch für den Fall, dass das Spiel abgebrochen wird.
Ausnahmen	IOException.

Tabelle 14: stopEngine Methode der ConnectionFacade Schnittstelle

Schnittstelle	ConnectionFacade
Methode	serializeDrawPDU
Eingabe	drawFlag: boolean
Ausgabe	/
Beschreibung	Sendet DrawPDU mit gegebener drawFlag an das verbundene Gerät.
Ausnahmen	Es kommt zu I/O-Exceptions.

Tabelle 15: serializeDrawPDU Methode der ConnectionFacade Schnittstelle

Schnittstelle	ConnectionFacade
Methode	serializePlayerIDPDU
Eingabe	ID: long
Ausgabe	/
Beschreibung	Sendet PlayerIDPDU mit gegebener ID an das verbundene Gerät.
Ausnahmen	Es kommt zu I/O-Exceptions.

Tabelle 16: serializePlayerIDPDU Methode der ConnectionFacade Schnittstelle

Schnittstelle	ConnectionFacade
Methode	serializeStartPDU
Eingabe	savedState: int[][]
Ausgabe	/
Beschreibung	Sendet StartPDU mit gegebenem State an das verbundene Gerät.
Ausnahmen	Es kommt zu I/O-Exceptions.

Tabelle 17: serializeStartPDU Methode der ConnectionFacade Schnittstelle

Schnittstelle	ConnectionFacade
Methode	serializeMovePDU
Eingabe	newState: int[][], endCondition: int
Ausgabe	/
Beschreibung	Sendet MovePDU mit gegebenen Parametern an das verbundene Gerät.
Ausnahmen	Es kommt zu I/O-Exceptions.

Tabelle 18: serializeMovePDU Methode der ConnectionFacade Schnittstelle

Schnittstelle	ConnectionFacade
Methode	serializeErrorPDU
Eingabe	type: String, message: String
Ausgabe	/
Beschreibung	Sendet ErrorPDU mit gegebenen Parametern an das verbundene Gerät.
Ausnahmen	Es kommt zu I/O-Exceptions.

Tabelle 19: serializeErrorPDU Methode der ConnectionFacade Schnittstelle

Schnittstelle	ConnectionFacade
Methode	unregisterReceiver
Eingabe	context: Context
Ausgabe	/
Beschreibung	Sorgt dafür, dass der Broadcast Receiver keine Updates zur Verbindungsänderung mehr an die Activity sendet.
Ausnahmen	Es ist kein Receiver registriert.

Tabelle 20: unregisterReceiver Methode der ConnectionFacade Schnittstelle

Schnittstelle	ConnectionFacade
Methode	connectTo
Eingabe	playerID: long
Ausgabe	/
Beschreibung	Verbindung wird zum Gerät mit gegebener playerID aufgebaut.
Ausnahmen	Es kommt zu I/O-Exceptions. Das Gerät mit gegebener ID existiert nicht.

Tabelle 21: connectTo Methode der ConnectionFacade Schnittstelle

Die Getter Operationen sind trivial und werden hier nicht weiter besprochen.

10 EndStates

Diese Komponente dient der Bluetooth Engine und der Game Logic, indem es diesen eine Enum für die unterschiedlichen End-States des Spiels zur Verfügung stellt. Darunter fallen zum Beispiel Draw by Dead Position, Draw by Repetition, Draw by Stalemate, Won, Lost und No End, falls keine Endkondition mit dem jeweiligen Halbzug erreicht wurde.

10.1 Schnittstellenmethoden

Die Schnittstelle beinhaltet nichts weiter als einen Getter und ist somit trivial. Sie bedarf keiner weiteren Beschreibung.

11 Integrations- und Systemtestkonzept

Hier soll es nun um das Integrations- und Systemtestkonzept für die Anbindung der Komponenten an die UI Komponente und die Anbindung der Subkomponenten der UI aneinander gehen. Dabei wird unter „korrektes Verhalten“ der Schnittstellenmethoden das Einhalten der Schnittstellenbeschreibungen gemeint, sowie ein robustes Verhalten bei den beschriebenen Ausnahmefällen. Tests können sich über mehrere Methoden ausstrecken. Diese werden erst in späteren Phasen klar benannt (Sobald die TDD beginnt).

11.1 GameLogic Integration

Methode (Schnittstelle)	Test	Bestanden (j), nicht bestanden (n), noch nicht geschrieben (nng)
initGame (FENGenerator)	Die Methode zeigt das korrekte Verhalten.	j (testInitGameWhite, testInitGameBlack)
getFEN (FENGenerator)	Die Methode zeigt das korrekte Verhalten.	j (testGetFEN)
getState (FENGenerator)	Die Methode zeigt das korrekte Verhalten.	j (testGetState, testGetStateWithPartialRow)
evaluate (GameRules)	Die Methode zeigt das korrekte Verhalten für alle Figurtypen und Farben, abseits der noch nicht zu implementierenden Regeln, wie Rochade, En Passant oder Promotion.	nng
evaluate (GameRules)	Die Methode braucht nicht länger als 0,5 Sekunden für die Evaluation.	nng

Tabelle 22: Integrationstestkonzept für die Anbindung der GameLogic Komponente an die UI

11.2 Persistence Integration

Methode (Schnittstelle)	Test	Bestanden (j), nicht bestanden (n), noch nicht geschrieben (nng)
read (Reader)	Das lesen funktioniert korrekt.	j (testReadEmtpyFile, testReadOneLine, testReadMultipleLines, testReadNonExistingFile)
saveGame (Writer)	Das schreiben funktioniert korrekt.	j (testSaveGameOneLine, testSaveGameMultipleLines, testSaveGameNonExistingFolder)
writeID (Writer)	Das schreiben funktioniert korrekt.	j (testWriteIDOneLine, testWriteIDNonExistingFolder)
delete (Deleter)	Die Löschung funktioniert korrekt.	j (testDeleterEmptyFile, testDeleterOneLine, testDeleterMultipleLines, testDeleterNonExistingFile)

Tabelle 23: Integrationstestkonzept für die Anbindung der Persistence Komponente an die UI

11.3 BluetoothEngine Integration

Methode (Schnittstelle)	Test	Bestanden (j), nicht bestanden (n), noch nicht geschrieben (nng)
startEngine (ConnectionFacade)	Die Methode zeigt das korrekte Verhalten.	nng
stopEngine (ConnectionFacade)	Die Methode zeigt das korrekte Verhalten.	nng
connectTo (ConnectionFacade)	Die Methode zeigt das korrekte Verhalten.	nng
serializePlayerIDPDU (ConnectionFacade)	Die Methode zeigt das korrekte Verhalten.	nng
serializeStartPDU (ConnectionFacade)	Die Methode zeigt das korrekte Verhalten.	nng
serializeMovePDU (ConnectionFacade)	Die Methode zeigt das korrekte Verhalten.	nng
serializeErrorPDU (ConnectionFacade)	Die Methode zeigt das korrekte Verhalten.	nng
serializeDrawPDU (ConnectionFacade)	Die Methode zeigt das korrekte Verhalten.	nng
unregisterReceiver (ConnectionFacade)	Die Methode zeigt das korrekte Verhalten.	nng
getInstance (ConnectionFacade)	Die Methode zeigt das korrekte Verhalten.	nng

Tabelle 24: Integrationstestkonzept für die Anbindung der BluetoothEngine Komponente an die UI

Die Bluetooth Geräte lassen sich mithilfe von Mockito simulieren, sodass eine Bluetooth Socket Verbindung zwischen den Mocks stattfinden kann, um die Verbindung zu testen.

11.4 UI Subkomponenten Integration

An dieser Stelle soll es genügen, im Rahmen eines Systemtests die korrekte Navigation zu überprüfen. Ein Systemtestkonzept folgt im nächsten Subabschnitt.

11.5 Systemtestkonzept

Mit trivialen Methoden sind hierbei alle Getter und Setter gemeint.

Test	Bestanden (j), nicht bestanden (n), noch nicht geschrieben (nng)
Beim fertiggestellten System soll es zu einem Durchgang kommen, der alle nicht-trivialen Schnittstellen Methoden überprüft.	nng (aber manuell getestet, jedoch mangelt es bei Spiellogik, Draw Requests und End Messages, sowie Broadcast Receiver)
Es wird ein Spiel gegen einen anderen, nicht-technik affinen, Spieler gespielt, um die Benutzbarkeit zu überprüfen. Dieser kann sie zwischen 1 und 5 für je sehr kompliziert und sehr einfach bewerten. Bestanden ist der Test bei einer 4.	nng

Tabelle 25: Systemtestkonzept

12 Unit-Test Konzept

Methode (Klasse)	Test	Bestanden (j), nicht bestanden (n), noch nicht geschrieben (nng)
handleInputStream (ConnectionEngine)	Es wird lokal eine Verbindung zwischen 2 Endgeräten mit Emulatoren und BluetoothSockets simuliert. Dadurch werden PDUs serialisiert und deserialisiert und können so einfach getestet werden.	nng

Tabelle 26: Unit-Test Konzept für die ConnectionEngine

Literatur

[1] Chess.com. (o. d.). chess.com - play chess online - free games. chess.com. https://chess.com/.