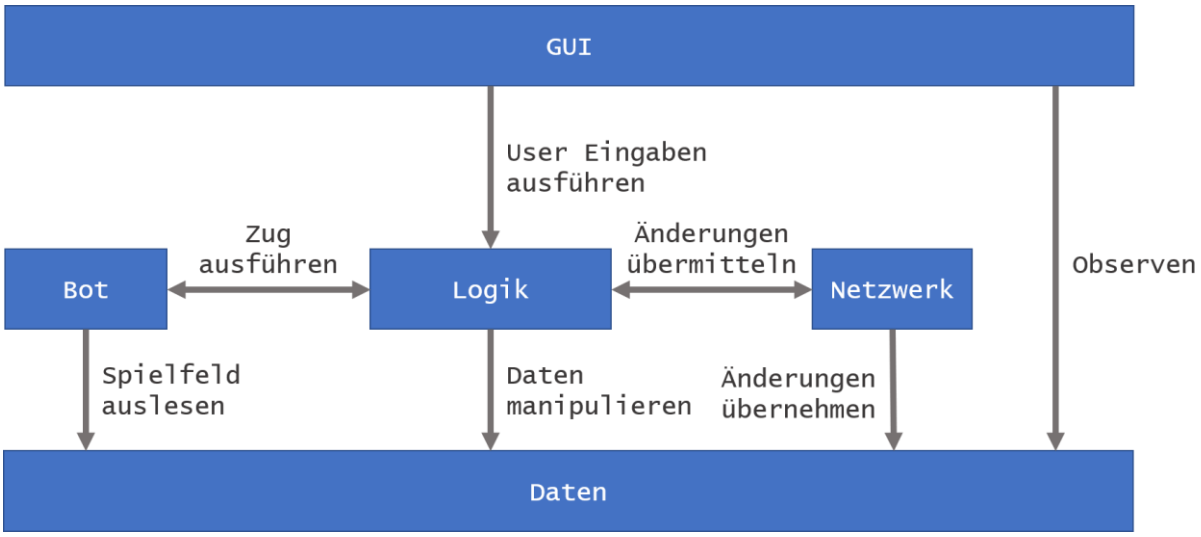
Architektur Dokumentation: Carcassonne

Das entwickelte Spiel ist in der Programmiersprache Java entwickelt. Für die Continuous Integration wurde die Gitlab CI genutzt, diese führt eine Reihe von Maven Befehlen aus, die erst das Projekt builded, testet und dann die JAR zum download bereitstellt. Ebenso wurden Codeanalysen durch SonarCube durchgeführt, welche Rückschlüsse auf mögliche Fehler oder unsaubere Programmierung geben kann.

Maven war ebenso für die Verwaltung der Dependencies verantwortlich, um auf allen Rechnern die gleichen Versionen bereit zu stellen. Für die graphische Oberfläche verwendeten wir JavaFX, zum Testen der App wurde JUnit verwendet. Ebenso nutzten wird Gson um Objekte in Json zu verwandeln und umgekehrt. Guava wurde als Unterstützung bei String Operationen verwendet.

Als Einstiegspunkt der App ist die Klasse de.btu.swp.carcassonne.App zu betrachten. In der Main-Methode startet JavaFX die App, in dem es von der aktuellen Klasse eine neue Instanz erstellt und diese initialisiert (init-Methode) und anschließend startet (start-Methode). Für die einzelnen Szenen werden FXML-Dateien geladen und diese werden dann mit ihren entsprechenden Controllern verbunden, welche dann für die weitere Steuerung der Szene verantwortlich sind.

Die GUI Controller haben eine Instanz der Logik (genannt CarcassonneService) mit welcher sie interagieren können. Der CarcassonneService besteht aus einzelnen Subservices, um die Aufgaben besser zu verteilen. Es gibt dann einen Verweis auf entsprechende Datenklassen, welche mit Properties arbeiten. Properties haben die Eigenschaft, dass sie über Listener Änderungen mitteilen und so Änderungen direkt an die GUI gehen können. Im groben Überblick sieht die Architektur dann so aus:



Das Netzwerk ist eigen entwickelt und basiert auf einer Server-Client-Architektur mit dem NIO Prinzip, dieses Prinzip wird durch die Standard Socket API von Java umgesetzt. Server und Client werden unter dem Interface Connection zusammengefasst, welches die Verwendung im ConnectionService ermöglicht. Daher ist es dem Spiel dann egal, ob es sich beim Verschicken der Pakets um einen Server handelt oder nicht. Jedes Paket weiß dann selbst, was für Änderungen es tätigen soll, in dem es eine Instanz des CarcassonneService übergeben bekommt.

Die Verbindung zwischen Client und Server basiert auf der IP und dem Port. Diese werden kodiert und können dann als Nutzer eingeben werden.