**Qualität**

Produktqualität

Ein Fokus bei der Entwicklung der Applikation zum Buchen von Kinotickets war die Qualität des Produktes selber, wie sie ein Endnutzer wahrnehmen wird. Der Endnutzer will eine einfache und intuitive Webseite besuchen, die visuell und auch von ihren Funktionen her viel zu bieten hat. Das Design der Webseite wurde mit Bedacht gewählt und angepasst, um einen objektiv schönen Anblick zu ermöglichen. Das Design ist auf der ganzen Seite konsistent und angenehm anzuschauen. Durch hohe Kontraste und Akzente bei den einzelnen Elementen auf der Seite ist es auch sinneseingeschränkten Benutzern der Webseite gestattet, diese zu ihrem vollen Ausmaß zu nutzen. Eine komplett barrierefreie Version ist für die Zukunft geplant, sodass dann auch zum Beispiel die Kompatibilität mit Hilfsmitteln wie Screenreadern optimiert wird. Dadurch, dass Menschen, die visuell eingeschränkt sind, nicht die Hauptzielgruppe eines Kinos sind, wurde dem Aspekt der Qualität bei der Entwicklung eine geringere Priorität gegeben.

**Technische Qualität**

Code Tests

Ein wichtiger Bestandteil von qualitativ hochwertigen Applikationen ist ein gut getesteter Code. Jede Funktion einer Applikation muss an sich funktionieren und darf auch nicht durch neue Änderungen oder anderen Funktionen in ihrer Essenz blockiert oder behindert werden. Um durchgehend im Rahmen von Continuous Integration sicherzustellen, dass jede Funktion im Code auch nach Änderungen noch weiterhin so funktioniert, wie sie es soll, werden Tests geschrieben. Diese Tests testen einmal alle Funktionen und Methoden an und für sich, ob sie – unabhängig von allen äußeren Einflüssen – im Grundsatz funktionieren. Zusätzlich zu diesen sogenannten Unit Tests werden Integrationstests ausgeführt, die das Zusammenspiel von verschiedenen Funktionen und Methoden untersuchen. Bei allen Tests wird eine Erwartung an die Funktion bzw. an das Zusammenspiel von Funktionen definiert, die erfüllt werden muss, damit die Applikation alle CI-Checks absolviert. Schlägt ein Test fehl kann das ein Anzeichen sein, dass die neu eingepflegte Funktion den Code an anderer Stelle einschränkt, was nicht gewollt und auch nicht gewünscht ist.

Die Effektivität und das Ausmaß der Tests kann durch die Code Coverage dargestellt werden. Die Code Coverage gibt an, wie viele Zeilen des Codes und wie viele verschiedene Abarbeitungswege im Code getestet werden. Eine hohe Code Coverage sagt an, dass die Chance Code von schlechter technischer Qualität herzustellen, deutlich geringer ist, als es bei einer niedrigen Code Coverage der Fall ist. Somit geht mit einer höheren Code Coverage auch eine höhere Qualität einher. Bei Applikation zum Buchen von Kinotickets wurde eine 100% Code Coverage angestrebt, die jede Zeile des Codes an jeder Stelle testet. Dies garantiert zwar noch keine absolute Ausfallsicherheit von Funktionen, ist aber wie bereits beschrieben ein sehr wichtiger Teil einer qualitativ hochwertigen Applikation.

(screenshot von code coverage am ende bei Abgabe)

Code Reviews

Ein weiterer Teil, der zu einer hohen Qualität beiträgt, sind Code Reviews, die durchgeführt werden. Auch wenn durch CI und CD alles automatisiert ist und diese Automatisierungen Sicherheiten geben, dass der Code und dadurch auch das Produkt eine hohe Qualität hat, ist es immer noch wichtig, mit eigenen Augen den Code anzuschauen. Auch wenn alle Checks und Tests im Rahmen von CI bestanden sind muss jedes Mal noch eine nicht an der Entwicklung der Funktion beteiligten Person den Code selber anschauen und bewerten. Fallen Unstimmigkeiten auf oder ist der Code an einigen Stellen zu kompliziert, ressourceneinnehmend oder nicht menschenlesbar, werden Änderungen beantragt. Somit werden nur Features und Funktionen in den Haupt-Code integriert, die nicht nur technisch gut, sondern auch optisch und durch menschliche Augen als qualitativ hochwertig abgesegnet sind.

**Hosting**

Das Backend wird auf Microsoft Azure gehostet. Microsoft Azure bietet Studentinnen und Studenten einen kostenlosen Account mit Startguthaben und Zugang zu vielen kostenlosen Services, die für die Entwicklung eines Backends benötigt werden. Azure stellt Webseiten bereit, auf denen das Backend deployed werden kann, sowie die nötigen Automatisierungsmechanismen, damit dies auch automatisch und einfach passieren kann. Zusätzlich stellt Azure auch Datenbanken zur Verfügung, von denen jedoch hier kein Gebrauch gemacht wird. Der Chatbot TheoTicket wird wie das Backend auf Microsoft Azure gehostet, da die gleiche Technologie wie im Backend genutzt wird.

Im Frontend wird durch einen anderen Einsatz von Technologien auf Microsoft Azure verzichtet, da das Hosten dort mit den Frontendtechnologien, die auf der Webseite genutzt werden, langsamer und störanfälliger ist, als es beim Backend der Fall ist. Deshalb wird das Frontend auf Vercel gehostet, welcher alle Anforderungen erfüllt. Auf Vercel kann das Frontend geladen und gehostet werden, nachdem auch hier automatische Prozesse zum deployen ablaufen, damit dies nicht von Hand passieren muss. Im Gegensatz zu Azure hat Vercel jedoch nicht so viele Überwachungs- und Konfigurationsfunktionen, jedoch sind alle Deployments stabil und funktioneren reibungslos.

**TheoTicket**

TheoTicket ist ein vom Backend unabhängiger Microservice, der die Funktionalitäten eines Chatbots besitzt. TheoTicket kann vom Endnutzer benutzt werden, um schnell Informationen zu finden, die auf der Webseite einige Klicks benötigen. Vor allem als Suchassistent ist TheoTicket gut zu gebrauchen, da er auf verschiedene Arten von Anfragen verschiedene Antworten hat. Dabei ist es egal, ob die Nachricht Rechtschreibfehler enthält oder umschrieben formuliert werden. Der Chatbot analysiert die Anfrage, gleicht sie mit der im Backend benutzten Datenbank ab und formuliert auf dessen Basis eine Antwort, die so gut wie möglich die Anfrage beantwortet. TheoTicket lernt nicht mit, sondern ist als statische Suchhilfe geplant und implementiert worden. Aus dem Grund sind keine alltäglichen Unterhaltungen möglich.