# Toolauswahl und Begründung
# für das Projekt
# „Entwicklung einer Litera Bücherkatalog-Anwendung“

**Bearbeitet von:**
Ranim Khallouf
Marija Ratnikova
Rama Alshaer
Lena Heberlein

**03. April 2025**

# Tool für Prototyping der Benutzerschnittstelle

| Gewichtung | Kriterium | Punkte(1-5) | Figma | Wireflame | GoMockingbird |
|---|---|---|---|---|---|
| 20% | Benutzerfreundlichkeit | 1 - nicht bequem<br>5 - sehr bequem | 5 | 4 | 4 |
| 15% | Funktionsumfang | 1 - sehr eingeschränkt<br>5 - sehr umfangreich | 5 | 2 | 3 |
| 7% | Zusammenarbeit | 1 - keine Kollaboration<br>5 - umfassende Echtzeit-Kollaboration | 5 | 2 | 4 |
| 23% | Preis | 1 - sehr teuer<br>5 - kostenlos/günstig | 4 | 4 | 4 |
| 10% | Popularität | 1 - kaum bekannt<br>5 - sehr beliebt, große Community | 5 | 3 | 2 |
| 25% | Team Erfahrung | 1 - keine Erfahrung<br>5 - sehr erfahren | 4 | 1 | 1 |
| **Auswahl und Gesamtpunkte** | | | **4.52** | 2,9 | 2,71 |

**Begründung:** Figma stellt im Vergleich zu anderen Prototyping-Tools die überzeugendste Lösung dar. Es bietet eine benutzerfreundliche und intuitiv bedienbare Oberfläche, umfangreiche Funktionen zur Erstellung interaktiver Prototypen sowie eine hervorragende Unterstützung für die Zusammenarbeit im Team in Echtzeit. Darüber hinaus ist das Tool weit verbreitet und verfügt über eine aktive Community sowie eine Vielzahl an Ressourcen. Da das Projektteam bereits mit Figma vertraut ist, kann der Einsatz ohne zusätzlichen Schulungsaufwand erfolgen. Insgesamt vereint Figma Effizienz, Funktionalität und Nutzerfreundlichkeit und ist somit die geeignetste Wahl für die Prototypenerstellung in diesem Projekt.

# IDE/Editor

| Gewichtung | Kriterium | Punkte(1-5) | Eclipse | IntelliJ IDEA | VS Code |
|---|---|---|---|---|---|
| 20% | Performance | 1 – sehr langsam<br>5 – sehr schnell | 3 | 4 | 5 |
| 15% | Erweiterbarkeit | 1 – kaum Plugins<br>5 – umfangreiche Plugin-/Erweiterungs-Ökosysteme | 5 | 5 | 5 |
| 20% | Benutzerfreundlichkeit | 1 – sehr kompliziert<br>5 – sehr intuitiv, leicht zu bedienen | 3 | 5 | 5 |
| 15% | Preis | 1 - sehr teuer<br>5 - kostenlos/günstig | 5 | 2 | 5 |
| 10% | Community Support | 1 – kaum bekannt, wenig Dokumentation<br>5 – sehr bekannt, großes Support-Netz | 4 | 5 | 5 |
| 25% | Team Erfahrung | 1 - keine Erfahrung<br>5 - sehr erfahren | 3 | 1 | 4 |
| **Auswahl und Gesamtpunkte** | | | 3.85 | 3.6 | **5** |

**Begründung:** Visual Studio Code (VS-Code) ist eine moderne, leistungsstarke und zugleich leichtgewichtige Entwicklungsumgebung, die sich ideal für unterschiedlichste Programmierprojekte eignet. Sie bietet eine hohe Benutzerfreundlichkeit, eine schnelle Performance und lässt sich durch eine Vielzahl von Erweiterungen flexibel an individuelle Anforderungen anpassen.
Ein wesentlicher Vorteil ist, dass VS Code kostenlos und plattformunabhängig verfügbar ist, wodurch es sowohl für Einzelentwickler als auch für größere Teams besonders attraktiv ist. Dank der starken Community und der umfassenden Dokumentation ist die Einarbeitung einfach, und bei Problemen stehen zahlreiche Ressourcen zur Verfügung. Darüber hinaus ist das Projektteam bereits mit VS Code vertraut, was eine effiziente und reibungslose Nutzung ohne zusätzlichen Schulungsaufwand ermöglicht. Aufgrund dieser Merkmale stellt VS Code eine äußerst geeignete Wahl für die Entwicklung in unserem Projekt dar.

# Test-Automatisierung

| Gewichtung | Kriterium | Punkte(1-5) | JUnit | TestNG | Selenium |
|---|---|---|---|---|---|
| 20% | Integration in IDE | 1 – kaum IDE-Integration<br>5 – nahtlose, erweiterte Integration | 5 | 5 | 5 |
| 20% | Dokumentation & Community | 1 – wenig Doku, kleine Community<br>5 – umfassende Doku, große Community | 4 | 4 | 5 |
| 25% | Lernkurve | 1 – sehr schwer zu erlernen<br>5 – sehr einfach | 4 | 3 | 3 |
| 15% | Plattformunabhängigkeit | 1 – eingeschränkt<br>5 – überall nutzbar | 5 | 5 | 5 |
| 10% | Funktionale Abdeckung | 1 – nur Unit-Tests<br>5 – Unit, Integration, E2E/GUI | 3 | 4 | 5 |
| 10% | Team Erfahrung | 1 - keine Erfahrung<br>5 - sehr erfahren | 1 | 1 | 1 |
| **Auswahl und Begründung** | | | **3.95** | 3.8 | 4.1 |

**Begründung:** JUnit ist die geeignetste Wahl für die Testautomatisierung. Es ist ein leichtgewichtiges, etabliertes Framework, das speziell für Unit-Tests entwickelt wurde und sich nahtlos in Visual Studio Code integrieren lässt.
Im Vergleich zu anderen Testwerkzeugen wie TestNG oder Selenium ist JUnit deutlich einfacher zu erlernen und anzuwenden, was eine schnelle und effektive Implementierung von Tests ermöglicht. Darüber hinaus überzeugt es durch klare Syntax, umfangreiche Dokumentation und eine große Entwickler-Community.

# Dokumentationstools

| Gewichtung | Kriterium | Punkte(1-5) | Javadoc | Doxygen | DocFX |
|---|---|---|---|---|---|
| 20% | Integration in Java-Projekte | 1 – kaum integrierbar<br>5 – vollständig integriert in Java und Build-Tools | 5 | 3 | 2 |
| 15% | Benutzerfreundlichkeit | 1 – unübersichtliche Oberfläche<br>5 – einfach zu bedienen | 5 | 3 | 2 |
| 10% | Formatvielfalt/Export | 1 – nur ein Format (z. B. HTML)<br>5 – viele Formate (HTML, PDF, Markdown, etc.) | 3 | 5 | 5 |
| 10% | Einrichtungsaufwand | 1 – sehr kompliziert<br>5 – einfach, keine zusätzliche Installation nötig | 5 | 3 | 3 |
| 15% | Funktionsumfang | 1 - sehr eingeschränkt<br>5 - sehr umfangreich | 5 | 2 | 3 |
| 5% | Preis | 1 - sehr teuer<br>5 - kostenlos/ Open Source | 5 | 5 | 3 |
| 5% | Popularität | 1 - kaum bekannt<br>5 - sehr bekannt | 5 | 2 | 1 |
| 20% | Team Erfahrung | 1 - keine Erfahrung<br>5 - gute Erfahrung | 3 | 1 | 1 |
| **Auswahl und Gesamtpunkte** | | | **4.40** | 2.70 | 2.35 |

**Begründung:** Nach einem umfassenden Vergleich der drei Dokumentationstools zeigt sich, dass Javadoc in den meisten Hauptkriterien überlegen ist. Es ist vollständig in die Java-Entwicklungsumgebung integriert und ein offizieller Bestandteil des JDK. Dadurch ist es besonders einfach zu verwenden und erfordert keine zusätzliche Installation oder Konfiguration. Obwohl das Team keine vorherige Erfahrung mit Javadoc hat, ist das Tool sehr einsteigerfreundlich und leicht verständlich. Darüber hinaus ist Javadoc vollständig kostenlos, was es auch unter wirtschaftlichen Aspekten attraktiv macht. Trotz der erweiterten Exportformate von Tools wie Doxygen und DocFX bleibt Javadoc aufgrund seiner Einfachheit, Java-Integration und geringen Einstiegshürde die beste Wahl für unser Projekt.

# Obfuscator

| Gewichtung | Kriterium | Punkte (1–5) | ProGuard | JavaGuard | DashO |
|---|---|---|---|---|---|
| 25% | Obfuskationsstärke | 1 – schwache Verschleierung<br>5 – sehr starke Verschleierung | 4 | 3 | 5 |
| 20% | Support für Java-Standards | 1 – veraltet, kein Support<br>5 – unterstützt neueste Standards | 5 | 3 | 5 |
| 15% | Lizenzmodell/Kosten | 1 – sehr teuer<br>5 – kostenlos / Open Source | 5 | 5 | 2 |
| 15% | Fehlertoleranz & Debugging | 1 – schwer nachvollziehbare Fehler<br>5 – klare Fehlermeldungen und Logs | 3 | 2 | 4 |
| 15% | Benutzerfreundlichkeit | 1 – kompliziert<br>5 – sehr intuitiv und benutzerfreundlich | 4 | 3 | 4 |
| 5% | Team Erfahrung | 1 – keine Erfahrung<br>5 – umfassende Erfahrung | 1 | 1 | 1 |
| 5% | Popularität | 1 – kaum bekannt<br>5 – sehr bekannt | 4 | 2 | 3 |
| **Auswahl und Gesamtpunkte** | | | **4.05** | 3.00 | 3.95 |

**Begründung**: Basierend auf der Auswertung in der Tabelle ist ProGuard die beste Wahl für unser Projekt. Es bietet eine gute Balance zwischen Sicherheit, moderner Java-Unterstützung und Benutzerfreundlichkeit. Als Open-Source-Tool ist es zudem kostenlos, was es besonders attraktiv für unser Budget macht. Im Vergleich dazu ist DashO zwar technisch stark, aber kostenpflichtig, während JavaGuard in wichtigen Kriterien wie Standardunterstützung und Bedienbarkeit schwächer abschneidet. Insgesamt erfüllt ProGuard unsere Anforderungen am besten und stellt somit die sinnvollste Lösung für unser Projekt dar.

# Codeconventions

| Gewichtung | Kriterium | Punkte(1-5) | Oracle Java Code Conventions | Google Java Style Guide | Mozilla Java Style Guide |
|---|---|---|---|---|---|
| 20% | Einrückung | 1 – unübersichtlich, inkonsistent<br>5 – klar definiert und einheitlich | 4 | 2 | 4 |
| 15% | Zeilenlänge | 1 – sehr kurz oder unpraktisch<br>5 – flexibel, aber gut lesbar | 3 | 5 | 4 |
| 15% | Klammern | 1 – uneinheitliche oder verwirrende Platzierung<br>5 – konsequent und leserlich | 3 | 5 | 5 |
| 10% | Variablennamen | 1 – unklare Konventionen<br>5 –einheitlich und verständlich | 5 | 5 | 5 |
| 15% | Importe | 1 - erlaubt unübersichtliche Wildcard-Importe<br>5 - erzwungene Klarheit durch explizite Importe | 2 | 5 | 5 |
| 20% | Team Erfahrung | 1 - keine Erfahrung<br>5 - gute Erfahrung | 1 | 1 | 1 |
| **Auswahl und Gesamtpunkte** | | | 2.75 | 3.55 | **3.95** |

**Begründung:** Wir haben uns für den Mozilla Java Style Guide (3,95 Punkte) entschieden, da er eine gute Balance zwischen Konsistenz und Flexibilität bietet. Die klare Struktur sorgt für eine einheitliche Code-Formatierung, während gleichzeitig genügend Freiraum für individuelle Anpassungen bleibt. Dadurch können wir unseren Code übersichtlich und gut lesbar gestalten, ohne uns zu stark an strikte Vorgaben halten zu müssen. Obwohl wir mit diesem Style Guide bisher noch keine Erfahrung haben, ist er leicht verständlich und gut dokumentiert, sodass eine Einarbeitung problemlos möglich ist. Zudem orientiert er sich an bewährten Standards, wodurch eine langfristige Wartbarkeit des Codes sichergestellt wird. Insgesamt bietet der Mozilla Style Guide die beste Mischung aus Praxisnähe, Lesbarkeit und Anpassungsfähigkeit für unser Projekt.

# Kollaborationstool

| Gewichtung | Kriterium | Punkte(1-5) | Mailingliste | Treffen | Issue Tracker |
|---|---|---|---|---|---|
| 20% | Kommunikationsform | 1 – langsam, schwer nachvollziehbar<br>5 – schnell und effizient | 3 | 5 | 4 |
| 20% | Dokumentation | 1 – keine oder unstrukturierte Nachverfolgung<br>5 – vollständige und strukturierte Speicherung | 4 | 1 | 5 |
| 15% | Interaktion | 1 – umständlich, wenig direkte Kommunikation<br>5 – sehr interaktiv und effizient | 2 | 5 | 3 |
| 15% | Effizienz | 1 – langsam, nicht zielgerichtet<br>5 – schnelle und direkte Lösungsmöglichkeiten | 3 | 5 | 5 |
| 10% | Eignung für Teamarbeit | 1 - ungeeignet für kooperative Aufgaben<br>5 - optimal für Teamarbeit | 4 | 4 | 5 |
| 20% | Team Erfahrung | 1 - keine Erfahrung<br>5 - gute Erfahrung | 1 | 5 | 1 |
| **Auswahl und Gesamtpunkte** | | | 2.65 | **4.45** | 3.55 |

**Begründung:** Wir haben uns für Treffen im Praktikum (4,45 Punkte) entschieden, da sie die effizienteste und direkteste Form der Kommunikation bieten. Durch den persönlichen Austausch können wir Probleme in Echtzeit besprechen und Missverständnisse vermeiden, was zu schnelleren und präziseren Lösungen führt. Zudem sind Treffen für uns besonders praktisch, da wir bereits Erfahrung damit haben und keine zusätzlichen Tools oder schriftlichen Dokumentationen erforderlich sind. Der direkte Kontakt fördert auch die Zusammenarbeit im Team, da Rückfragen sofort geklärt werden können und Diskussionen dynamischer verlaufen. In einem Praktikumsumfeld, in dem schnelle Abstimmung und unkomplizierte Absprachen entscheidend sind, sind persönliche Treffen daher die beste Wahl für unsere Bedürfnisse.

# Versionsverwaltungssysteme

| Gewichtung | Kriterium | Punkte(1-5) | hg | Git | SVN |
|---|---|---|---|---|---|
| 20% | Benutzerfreundlichkeit | 1 - nicht bequem<br>5 - sehr bequem | 4 | 4 | 4 |
| 15% | Verbreitung /Community | 1 - kaum bekannt<br>5 - sehr verbreitet | 3 | 5 | 4 |
| 15% | Tool-/IDE-Integration | 1 - schlecht integriert<br>5 - sehr gut integriert | 3 | 5 | 4 |
| 15% | Offline-Nutzung | 1 - eingeschränkt<br>5 - vollständig offlinefähig | 5 | 5 | 2 |
| 10% | Branching & Merging | 1 - schlecht<br>5 - sehr flexibel | 4 | 5 | 2 |
| 25% | Team Erfahrung | 1 - keine Erfahrung<br>5 - sehr erfahren | 1 | 4 | 1 |
| **Auswahl und Gesamtpunkte** | | | 3.10 | **4.3** | 2,75 |

**Begründung:** Git bietet moderne und flexible Arbeitsweisen, ist weit verbreitet und hervorragend in Entwickler-Tools wie GitHub, GitLab oder VS Code integriert. Es unterstützt verteilte Teams, ermöglicht effizientes Branching & Merging und wird durch eine große Community getragen. Da das Projektteam bereits erfahren im Umgang mit Git ist, kann das Tool ohne zusätzlichen Aufwand eingesetzt werden. Insgesamt ist Git damit die beste Wahl für eine effektive und reibungslose Zusammenarbeit im Projekt.

# UML-Tools mit Codegenerierung

| Gewichtung | Kriterium | Punkte(1-5) | Papyrus | StarUML | PlantUML |
|---|---|---|---|---|---|
| 10% | Codegenerierung | 1 - schwach<br>5 - sehr gut integriert | 4 | 3 | 1 |
| 20% | Benutzerfreundlichkeit | 1 - technisch<br>5 - sehr intuitiv | 3 | 5 | 4 |
| 20% | Tool-/IDE-Integration | 1 - begrenzt<br>5 - vollständig integriert | 2 | 5 | 4 |
| 20% | Open Source / Lizenz | 1 - proprietär<br>5 - kostenlos/offen | 5 | 3 | 4 |
| 10% | Aktualisierungen&Community | 1 - kaum aktiv<br>5 - regelmäßige Updates & große Community | 4 | 4 | 4 |
| 20% | Team Erfahrung | 1 - keine Erfahrung<br>5 - sehr erfahren | 3 | 1 | 4 |
| **Auswahl und Gesamtpunkte** | | | 3.40 | 3.50 | **3.70** |

**Begründung:** PlantUML erreicht in der Bewertung die höchste Gesamtpunktzahl und bietet eine starke Kombination aus Funktionalität, Offenheit und praktischer Einsetzbarkeit. Besonders hervorzuheben ist die sehr gute Open-Source-Lizenz, die eine kostenlose und flexible Nutzung ohne Einschränkungen erlaubt. Die textbasierte Modellierung ermöglicht eine einfache Integration in bestehende Entwicklungs- und Versionskontrollprozesse, was gerade für agile Teams von Vorteil ist.
Zudem verfügt das Projektteam über solide Erfahrung im Umgang mit PlantUML, wodurch sich das Tool ohne Schulungsaufwand direkt produktiv einsetzen lässt.

# Build-Tools

| Gewichtung | Kriterium | Punkte(1-5) | Maven | Gradle | Ant |
|---|---|---|---|---|---|
| 25% | Einfachheit /Lernkurze | 1 – schwer verständlich<br>5 – schnell & intuitiv erlernbar | 3 | 5 | 4 |
| 20% | Dokumentation | 1 – keine oder unstrukturierte Hilfe<br>5 – umfangreiche, gut strukturierte Doku | 5 | 4 | 2 |
| 20% | DIE-Integration | 1 – kaum unterstützt<br>5 – nahtlos integriert | 5 | 5 | 3 |
| 15% | Projektstruktur&Klarheit | 1 – unklar<br>5 – klar definierte (Ordner-/Ablaufstruktur) | 5 | 4 | 2 |
| 15% | Erweiterbarkeit | 1 – eingeschränkte Erweiterung<br>5 – offen und anpassungsfähig | 4 | 5 | 2 |
| 5% | Team Erfahrung | 1 - keine Erfahrung<br>5 - gute Erfahrung | 1 | 1 | 1 |
| **Auswahl und Gesamtpunkte** | | | **4.45** | 4.15 | 2.65 |

**Begründung:** Maven bietet eine ausgewogene Kombination aus Struktur, Stabilität und Benutzerfreundlichkeit, die besonders für studentische Projekte vorteilhaft ist. Die klare Projektstruktur erleichtert den Einstieg und sorgt für eine saubere Organisation des Codes.

Dank der umfangreichen Dokumentation, einer aktiven Community sowie der nahtlosen Integration in gängige IDEs  lässt sich Maven auch ohne Vorkenntnisse effizient nutzen.