






**CODESPHERE SOLUTIONS**  
TRANSFORMING VISIONS INTO CODE

## Toolvergleich

### Versionierung

| Kriterium                  | Git   | Mercurial  | Subversion   |
|----------------------------|---|--|--|
| <b>Beschreibung</b>        |  <p><b>Git</b> ist ein verteiltes Versionskontrollsystem zur Nachverfolgung von Änderungen an Dateien.</p> |  <p><b>Mercurial</b> ist ein Versionskontrollsystem, das auf Leistung, Skalierbarkeit und einfache Bedienung ausgelegt ist.</p> |  <p><b>Subversion (SVN)</b> ist ein zentrales Versionskontrollsystem zur Verwaltung von Änderungen an Dateien und Verzeichnissen.</p> |
| <b>Market Share</b>        | 87,50%  | 0,10%  | 2,20%  |
| <b>Einrichtung</b>         | Mittel, viele Konfigurationsoptionen  | Einfach und benutzerfreundlich   | Einfach, aber zentraler Server benötigt  |
| <b>Komplexität</b>         | Komplex, da viele Optionen  | Einfache und konsistente Syntax  | Einfach für Einsteiger, wenige Befehle   |
| <b>Dezentrale Arbeit</b>   | Ja, alle Entwickler haben ein Repository  | Ja, alle Entwickler haben ein Repository   | Nein, zentraler Server verwaltet alle Versionen  |
| <b>Hosting-Plattformen</b> | GitHub, GitLab, Azure DevOps  | Bitbucket  | Apache SVN, Assembla   |
| <b>Schwerpunkte</b>        | Flexibilität  | Effizienz, Skalierbarkeit und robuste Handhabung von Text- und Binärdateien  | Atomare Operationen (sichere Ausführung)   |
| <b>Anmerkungen</b>         | begrenzte Unterstützung für Dateisperren, ineffiziente Diffs und Merge-Konflikte, machen Git für große Binärdateien weniger geeignet.   | Eclipse Integration ist für Mercurial nicht so gut wie bei den anderen Optionen  | Skalierbarkeitsprobleme bei großen Projekten   |




#### Auswahlbegründung:

Obwohl Subversion und Mercurial leichter zu erlernen sind, ist **Git** flexibler anwendbar und durch die größere Community sind Lösungen für mögliche Probleme einfacher aufzufinden.



**CODESPHERE SOLUTIONS**  
TRANSFORMING VISIONS INTO CODE

## UML-Tool mit Quellcodegenerierung

| Kriterium                             | Papyrus (Eclipse)  | StarUML  | Visual Paradigm (Visual Paradigm for UML)   |
|---------------------------------------|--|--|---|
| <b>Beschreibung</b>                   |  <p>Grafisches Bearbeitungstool für UML2 nach OMG</p> |  <p>UML-Tool, mit Ziel, große, kommerzielle Applikationen zu ersetzen</p> |  <p>Desktop-Software zur Modellierung von Software mit UML und SysML Unterstützung</p> |
| <b>Unterstützte Sprachen</b>          | Java, C++, Python (via Plugins)  | Java, C++, C#, Python, PHP   | Java, C++, C#, PHP  |
| <b>Plattform</b>                      | Windows, Linux, MacOS  | Windows, Linux, MacOS  | Windows, Linux, MacOS   |
| <b>Qualität des generierten Codes</b> | Mittel   | Mittel   | sehr hoch   |
| <b>Lizenz</b>                         | Eclipse Public License, Open Source  | modifizierte GNU General Public License, Freemium  | proprietär, Kommerziell   |
| <b>Besondere Features</b>             | Starke Eclipse-Integration, Modellgetriebenes Engineering  | Erweiterbar mit Plugins, schlankes UI  | Code-Generierung und Reverse-Engineering, BPMN Unterstützung  |
| <b>Zielgruppe</b>                     | Entwickler mit Eclipse-Umgebung  | Entwickler, die ein leichtes UML-Tool suchen   | Unternehmen und professionelle Modellierung   |
| <b>Kosten</b>                         | kostenlos  | Standard: 129\$/user one-time<br>Professional: 199\$/user one-time   | Modeler: 99\$<br>Standard: 349\$<br>Professional: 799\$<br>Enterprise: 1999\$   |




### Auswahlbegründung:

**Papyrus** scheint mit seiner direkten Eclipse-Einbindung das Tool der Wahl zu sein. Visual Paradigms hohe Kosten sprechen trotz der sehr guten Code-Generierung gegen das Tool. Auch StarUML verliert in der Gegenüberstellung gegen Papyrus, das frei nutzbar ist und keine Kosten verursacht.



**CODESPHERE SOLUTIONS**  
TRANSFORMING VISIONS INTO CODE

## Build-Tool

| Kriterium                      | Gradle  | Maven  | Ant   |
|--------------------------------|---|--|---|
| <b>Beschreibung</b>            |  <p><b>Gradle</b> ist ein flexibles Build-Tool mit deklarativer DSL und imperativer Logik, ideal für komplexe Abhängigkeiten in Android- und Java-Projekten.</p> |  <p><b>Maven</b> ist ein XML-basiertes Build-Tool für Java, nutzt zentrale Repositories und Konventionen, ideal für große, standardisierte Projekte.</p> |  <p><b>Apache Ant</b> ist ein XML-basiertes Build-Tool, flexibler als Maven, benötigt explizite Build-Skripte und eignet sich für angepasste Prozesse.</p> |
| <b>Build-Systemtyp</b>         | Imperativ und deklarativ (DSL)  | Deklarativ (XML)   | Imperativ (XML)   |
| <b>Build-Skript</b>            | Groovy/Kotlin DSL<br>- flexibel, lesbar, programmierbar   | XML (pom.xml)<br>- Starrer aber standardisiert   | XML (build.xml)<br>- Sehr manuell, keine vordefinierten Konventionen  |
| <b>Abhängigkeitsmanagement</b> | Integriert (Maven-Repository)<br>- Transitive Dependencies automatisch  | Integriert (pom.xml)<br>- Transitive Dependencies, zentrale Verwaltung   | Manuell (kein natives Management)<br>- Manuelles Herunterladen oder Ivy-Plugin  |
| <b>Performance</b>             | Schnell<br>(Inkrementelle Builds, Caching)  | Langsamer<br>(starre Struktur, kein Cache für wiederholte Tasks)   | Langsam<br>(kein Dependency-Management, manuelle Task-Ausführung)   |
| <b>Flexibilität</b>            | Sehr flexibel<br>(eigene Tasks, Plugins, Skriptlogik)   | Weniger flexibel<br>(vordefinierte Phasen und Standards)   | Sehr flexibel<br>(Buildschritte können manuell definiert werden)  |
| <b>Plugins</b>                 | Große Plugin Community  | Viele Standard-Plugins   | Plugins verfügbar, aber weniger verbreitet  |
| <b>Lernkurve</b>               | Mittel (DSL erforderlich)   | Einfach (standardisierte Struktur)   | Hoch (manuelle Konfiguration)   |
| <b>Standardisierung</b>        | Keine strikten Konventionen   | Stark standardisiert (Maven-Konventionen)  | Keine Standardkonventionen  |
| <b>Multiprojekt-Builds</b>     | Gut unterstützt   | Unterstützt, aber komplex  | Manuell, aufwendig  |
| <b>IDE-Unterstützung</b>       | sehr gut (IntelliJ, Eclipse, Android Studio)  | sehr gut (Standard in Java IDEs)   | gut, aber veraltet (meist legacy Projekte)  |




### Auswahlbegründung:

Durch die sehr flexible Arbeit, schnelle Performance, einer nicht zu hohen Lernkurve, einer großen Unterstützung für Plugins und einer sehr guten IDE-Unterstützung bietet **Gradle** die beste Mischung aller Kriterien.



**CODESPHERE SOLUTIONS**  
TRANSFORMING VISIONS INTO CODE

## Tool für Prototyping der Benutzerschnittstelle

| Kriterium                          | Figma  | Adobe XD  | Sketch   |
|------------------------------------|--|---|--|
| <b>Beschreibung</b>                |  <p>Ein webbasiertes Design- und Prototyping-Tool, das für seine Echtzeit-Kollaborationsfunktionen bekannt ist.</p> |  <p>Ein benutzerfreundliches Tool von Adobe für UI/UX-Design und Prototyping mit nahtloser Integration in andere Adobe-Produkte. Besonders gut für interaktive Designs geeignet.</p> |  <p>Ein vektorbasiertes Design-Tool für macOS, das sich auf UI/UX-Design spezialisiert hat. Besonders beliebt bei Designern aufgrund seiner umfangreichen Plugins und einfachen Bedienung.</p>  |
| <b>Plattform</b>                   | Webbasiert (Windows, macOS, Linux)   | Windows, macOS  | Nur macOS  |
| <b>Offline-Nutzung</b>             | Eingeschränkt, benötigt Internet   | Ja  | Ja   |
| <b>Kollaboration</b>               | Echtzeit-Zusammenarbeit (Cloud-basiert)  | Eingeschränkte Echtzeit-Zusammenarbeit (Coediting)  | Kein natives Echtzeit-Coediting, aber mit Plugins möglich  |
| <b>Prototyping</b>                 | Integriert mit Interaktionen   | Integriert mit Auto-Animate   | Integriert, aber weniger flexibel  |
| <b>Design-Funktionen</b>           | Sehr umfangreich, viele Plugins  | Gut integriert in Adobe-Ökosystem   | Stark, aber abhängig von Plugins   |
| <b>Plugins &amp; Erweiterungen</b> | Viele Plugins, API für eigene Erweiterungen  | Gute Plugin-Unterstützung   | Sehr viele Plugins, stark erweiterbar  |
| <b>Dateiformat</b>                 | .fig-Dateien<br><br>Export: SVG, PNG, PDF, CSS-Code  | .xd-Dateien<br><br>Export: PNG, SVG, PDF, HTML & CSS-Code (mit Plugins möglich)   | .sketch-Dateien<br><br>Export: PNG, SVG, PDF, HTML & CSS-Code  |
| <b>Benutzerfreundlichkeit</b>      | Sehr intuitiv, moderne Oberfläche, einfach für Einsteiger  | Sehr einfach zu bedienen, besonders für Adobe-Nutzer  | Intuitiv, aber eher für macOS-User optimiert   |
| <b>Lernkurve</b>                   | Einfach zu erlernen  | Einfach zu erlernen   | Etwas komplexer  |
| <b>Ideal für</b>                   | Teams, Remote-Work, Startups   | Adobe-Nutzer, Prototyping   | macOS-Nutzer, Agenturen  |
| <b>Kosten</b>                      | <p>Basis Funktionen: <b>Kostenlos</b></p> <p>Erweiterte Funktionen: verschiedene Abo-Modelle je nach Anforderungen <b>von 3€ bis 90€</b> pro Monat</p>   | <p>Im Adobe CC-Abo enthalten (Alle Adobe Applikationen)</p> <p><b>33,21€ pro Monat.</b></p>   | <p>30-tägige kostenlose Testversion</p> <p>Einmalige Zahlung <b>\$120 Pro Platz</b> (Beinhaltet 1 Jahr lang Updates)</p> <p>Standardabonnement <b>\$10 pro Monat pro Redakteur</b> (jährliche Abrechnung möglich --&gt; spart 24\$)</p> <p>Geschäftsabonnement <b>\$22 pro Monat pro Redakteur</b> (Nur mit jährlicher Abrechnung verfügbar)</p> |




### Auswahlbegründung:

Aufgrund der aktuellen Situation, dass Adobe XD sich im Wartungsmodus befindet und nicht weiterentwickelt wird, sowie der Tatsache, dass Sketch nur für macOS-Nutzer verfügbar und relativ teuer ist, haben wir uns für **Figma** entschieden. Figma bietet eine plattformunabhängige Lösung, die sowohl auf Windows als auch auf macOS funktioniert und keine Einschränkungen hinsichtlich der Systemumgebung hat. Besonders hervorzuheben ist die Möglichkeit der Echtzeit-Kollaboration, die die Zusammenarbeit im Team deutlich erleichtert und die Produktivität steigert. Figma ist zudem cloud-basiert, was den Zugriff auf Projekte von überall ermöglicht und eine kontinuierliche Versionierung sicherstellt. Mit einem flexiblen Preismodell, das auch eine kostenlose Version für kleinere Teams bietet, ist Figma sowohl kostengünstig als auch zukunftssicher. Da das Tool kontinuierlich weiterentwickelt wird und die umfangreiche Plugin- und Integrationsmöglichkeiten bietet, stellt es eine langfristige und skalierbare Lösung für unsere Bedürfnisse dar.



**CODESPHERE SOLUTIONS**  
TRANSFORMING VISIONS INTO CODE

## IDE/Editor

| Kriterium                             | Eclipse  | IntelliJ  | NetBeans  |
|---------------------------------------|--|---|---|
| <b>Beschreibung</b>                   |  <p>Eine leistungsstarke, Open-Source-IDE, die durch zahlreiche Plugins stark erweiterbar ist. Besonders geeignet für große Projekte, unterstützt mehrere Programmiersprachen und bietet eine breite Community.</p> |  <p>Eine moderne und intuitive IDE von JetBrains mit herausragender Code-Analyse und Refactoring-Tools. Die Community Edition ist kostenlos, während die Ultimate Edition kostenpflichtig ist und erweiterte Features für professionelle Entwicklung bietet.</p> |  <p>Eine benutzerfreundliche IDE, die besonders für Einsteiger geeignet ist. Sie bietet integrierte GUI-Entwicklung für Swing und JavaFX, ist jedoch weniger flexibel als Eclipse oder IntelliJ.</p> |
| <b>Lizenz</b>                         | Open Source (EPL)  | Community: Open Source (Apache 2.0), Ultimate: Kommerziell  | Open Source (Apache 2.0)  |
| <b>Benutzerfreundlichkeit</b>         | Anfangs komplex, anpassbar   | Intuitiv, viele Features "out of the box"   | Einfach zu bedienen, weniger anpassbar  |
| <b>Leistung</b>                       | Langsam beim Start, ressourcenhungrig  | Schnell, optimierte Performance   | Solide, aber nicht so schnell wie IntelliJ  |
| <b>Plugins/Erweiterbarkeit</b>        | Sehr viele Plugins verfügbar   | Viele Plugins, aber weniger als Eclipse   | Weniger Plugins als Eclipse und IntelliJ  |
| <b>Unterstützte Sprachen</b>          | Java, C/C++, Python, PHP, u.a.   | Java, Kotlin, Groovy, Scala, u.a.   | Java, PHP, HTML, JavaScript, u.a.   |
| <b>Refactoring &amp; Code-Analyse</b> | Gut, aber weniger intuitiv   | Sehr leistungsfähig, KI-gestützt  | Solide, aber nicht so leistungstark wie IntelliJ  |
| <b>Integrierte Werkzeuge</b>          | Viele Tools über Plugins   | Umfangreiche Features von Haus aus  | Gute GUI-Builder-Integration  |
| <b>GUI-Designer</b>                   | Über Plugins möglich (z. B. WindowBuilder)   | Nicht enthalten (Plugins nötig)   | Integrierter GUI-Builder für Swing  |
| <b>Preis</b>                          | Kostenlos  | Community Edition: Kostenlos, Ultimate: Kostenpflichtig   | Kostenlos   |




### Auswahlbegründung:

Unser Entwicklerteam hat sich für **Eclipse** entschieden, da es als kostenlose Open-Source-IDE eine hohe Flexibilität und Erweiterbarkeit durch zahlreiche Plugins bietet. Im Vergleich zu IntelliJ, das in der Vollversion kostenpflichtig ist, und NetBeans, das weniger anpassbar ist, überzeugt Eclipse durch starke Performance, breite Community-Unterstützung und optimale Eignung für große Projekte.



**CODESPHERE SOLUTIONS**  
TRANSFORMING VISIONS INTO CODE

## Test-Automatisierung

| Kriterium                                | JUnit   | TestNG  | NUnit   |
|--|---|---|---|
| <b>Beschreibung</b>                      | <br>Framework zum Testen von Java-Programmen, baut auf SUnit für Smalltalk auf, Ursprung von xUnit | <br>Framework zum Testen von Java-Programmen, besonders automatisiert. Erweitert JUnit und NUnit, ergänzt diese aber | <br>Framework für Unit-Tests für .NET-Sprachen |
| <b>Unterstützte Sprachen</b>             | Java  | Java  | C# (.NET)   |
| <b>Test-Typen</b>                        | Unit-Tests, Integrations-Tests  | Unit-Tests, Integrations-Tests, parallele Tests   | Unit-Tests, Integration-Tests   |
| <b>Parallele Testausführung</b>          | Nein  | Ja  | Ja  |
| <b>Abhängigkeiten verwalten</b>          | Manuell (Setup/Teardown)  | Automatisch mit flexiblen Annotations   | Manuell mit Setup/Teardown  |
| <b>Anmerkungen (Annotations)</b>         | @Test, @BeforeEach, @AfterEach  | @Test, @BeforeSuite, @AfterSuite (mehr Optionen)  | [Test], [SetUp], [TearDown]   |
| <b>Berichts- und Logging-Integration</b> | Grundlegend   | Erweitert   | Mittel  |
| <b>Lizenz</b>                            | Eclipse Public License 2.0, Open Source   | Apache Lizenz 2.0, Open Source  | Open Source   |
| <b>Zielgruppe</b>                        | Standard-Java-Unit-Tests  | Komplexe Testszenarien mit parallelen Ausführungen  | .NET-Entwickler mit NUnit-Unterstützung   |



### Auswahlbegründung:

Unser Team wird sich für **JUnit** entscheiden, da TestNG zwar eine Weiterentwicklung ist, wir diese aber nicht benötigen. Außerdem ist JUnit nativ in Eclipse unterstützt. NUnit fällt aus der Auswahl heraus, da es für unsere Java-Anwendung nicht nutzbar ist.



**CODESPHERE SOLUTIONS**  
TRANSFORMING VISIONS INTO CODE

## Dokumentationstool

| Kriterium                    | Javadoc  | Doxygen   | GoogleDocs  |
|------------------------------|--|---|---|
| <b>Beschreibung</b>          |  <p>Ein in Eclipse integriertes Tool zur automatischen Erzeugung einer Dokumentation aus dem Quellcode heraus. Teilweise als "Standardwerkzeug" der Java-Dokumentaion betitelt.</p> |  <p>Ähnlich zu Javadoc. Unterstützt mehrere Programmiersprachen. Kann in Eclipse durch Plug-Ins eingebunden werden.</p> |  <p>Manuelles Erstellen der Dokumentation innerhalb des Projekt-Wikis.</p> |
| <b>Kosten</b>                | keine  | keine   | ab 6,80€ pro Nutzer, pro Monat  |
| <b>Lizenz</b>                | GNU GPL v2, unproblematisch  | GNU GPL v2, unproblematisch (gilt nur für erstellte Dokumentation)  | unproblematisch   |
| <b>Market Share</b>          | Hoch, wird als "Standardwerkzeug" bezeichnet   | Hoch, auch für andere Programmiersprachen verwendbar  | N/A   |
| <b>Mitarbeitern bekannt</b>  | nein   | nein  | teilweise   |
| <b>Automatisierungsgrad</b>  | automatische Erzeugung eines Wikis aus Quellcode und Kommentaren   | Ähnlich Javadoc, zusätzlich teilweise UML-Diagramm-Erzeugung  | Manuelles Erstellen der Dokumentation innerhalb des Projekt-Wikis.  |
| <b>Einbindung in Eclipse</b> | standardmäßig vorinstalliert in Eclipse  | durch Plug-Ins  | keine   |
| <b>Testergebnis</b>          | Simplel, einfach zu erlernen   | Ähnlich zu Javadoc. Einbindung in Eclipse gestaltet sich schwieriger. Scheint den Aufwand nicht wert zu sein, nur um Klassendiagramme automatisch erstellen zu lassen.                                    | Viel zu aufwändig   |

### Auswahlbegründung:

**Javadoc** scheint vorerst das geeignetste Dokumentationstool zu sein. Dies geht aus dem Vergleichstest hervor, da man es im Test einfacher nutzen konnte als Doxygen und Javadoc in seiner Funktionalität ausreicht (gegenüber der leicht erweiterten Funktionalität von Doxygen). Sollte unser Unternehmen in Erwägung ziehen, in Zukunft auf andere/weitere Programmiersprachen als Java umzusteigen, so sollte die Wahl des Dokumentationstools noch einmal überdacht werden, da man Doxygen programmiersprachenübergreifend nutzen kann. GoogleDocs entfällt aufgrund des miserablen Tabellenvergleichs.



**CODESPHERE SOLUTIONS**  
TRANSFORMING VISIONS INTO CODE

## Obfuscator

| Kriterium                         | ProGuard   | DashO  | Allatori  |
|-----------------------------------|--|--|---|
| <b>Beschreibung</b>               | <br>Freie Software mit grundlegendem Schutz | <br>Obfuscator für Anwendungen mit hohen Sicherheitsstandards | <br>Software mit erhöhtem Reverse Engineering Schutz |
| <b>Obfuscationstechniken</b>      | Name-Mangling, Dead-Code-Elimination, Shrinking, Control-Flow-Obfuscation  | Name-Mangling, String-Verschlüsselung, Control-Flow-Manipulation, Anti-Debugging, Native Code Protection                                       | Name-Mangling, Control-Flow-Obfuscation, Stringverschlüsselung  |
| <b>Integrationsaufwand</b>        | Gering, gut dokumentierte Einrichtung  | Hoch, detaillierte Konfiguration nötig   | Mittel, einige erweiterte Einstellungen nötig   |
| <b>Performancebeeinflussung</b>   | Sehr gering  | Mittel   | Hoch  |
| <b>Code-Wiederherstellbarkeit</b> | Leicht zu deobfusken   | Schwer zu deobfusken   | Schwer zu deobfusken  |
| <b>Besondere Features</b>         | Integriert in Android SDK  | Starke Anti-Tampering-Methoden   | Wasserzeichentechnologie zur Urheberrechtsverfolgung  |
| <b>Lizenz</b>                     | Open Source  | Kommerziell  | Kommerziell   |
| <b>Kosten</b>                     | Kostenlos  | Kostenpflichtig  | Kostenpflichtig   |

### Auswahlbegründung:




Wir haben uns für **ProGuard** entschieden, da die Anwendung im Café keine besonderen Anforderungen an die Sicherheit oder Datenverschleierung stellt. Es handelt sich um eine kostenfreie und leicht zu integrierende Methode und eignet sich daher am besten.





**CODESPHERE SOLUTIONS**  
TRANSFORMING VISIONS INTO CODE

## Codeconventions

| Kriterium                            | Oracle  | Google  | Mozilla  |
|--------------------------------------|---|---|--|
| <b>Beschreibung</b>                  |  <p><b>Oracle</b> legt Java-Richtlinien fest: CamelCase, vier Leerzeichen Einrückung, klare Klammern und Kommentare, verbreitet in Enterprise-Anwendungen.</p> |  <p><b>Google</b> legt strikte Formatierungsregeln fest mit 100 Zeichen pro Zeile, 2 Leerzeichen Einrückung, strukturierten Imports und klarer Lesbarkeit.</p> |  <p><b>Mozilla</b> setzt Code-Standards für Sicherheit und Performance, bevorzugt Smart Pointer in C++ und strenge Linter-Regeln in JavaScript.</p> |
| <b>Namenskonventionen</b>            | Klassen: PascalCase<br>Methoden: camelCase<br>Variablen: camelCase<br>Konstanten: UPPER_CASE  | Gleich wie Oracle, aber keine Unterstriche in Methodennamen (getJSON() statt get_JSON())  | Gleich wie Oracle  |
| <b>Einrückung &amp; Formatierung</b> | 4 Leerzeichen, keine Tabs   | 2 Leerzeichen, keine Tabs   | 4 Leerzeichen, keine Tabs  |
| <b>Maximale Zeilenlänge</b>          | 80 Zeichen empfohlen, max. 120 erlaubt  | 100 Zeichen empfohlen   | 99 Zeichen empfohlen   |
| <b>Klammern</b>                      | { in der gleichen Zeile (if (...) { ... })  | { in der gleichen Zeile   | { in der gleichen Zeile  |
| <b>Leerzeichen</b>                   | Vor und nach Operatoren (a + b, a ==)   | Gleich wie Oracle   | Gleich wie Oracle  |
| <b>Methodenlänge</b>                 | Möglichst kurz (< 50 Zeilen empfohlen)  | Möglichst kurz (< 40 Zeilen empfohlen)  | Möglichst kurz   |
| <b>Kommentare</b>                    | Javadoc für öffentliche Methoden/Klassen  | Javadoc für öffentliche Methoden/Klassen, // für interne Kommentare   | Javadoc für öffentliche Methoden/Klassen   |
| <b>Codeorganisation</b>              | Eine Klasse pro Datei   | Eine Klasse pro Datei, aber Enums dürfen mehrere enthalten  | Eine Klasse pro Datei  |
| <b>Fehlermeldungen</b>               | Aussagekräftige Exceptions  | Aussagekräftige Exceptions  | Aussagekräftig   |

### Auswahlbegründung:

Wir haben uns für den **Oracle Code Style Guide** entschieden, da er der offizielle Standard für Java ist und weit verbreitet wird. Er bietet eine ausgewogene Balance zwischen Lesbarkeit und Flexibilität, was ihn ideal für konsistenten und wartbaren Code macht.



**CODESPHERE SOLUTIONS**  
TRANSFORMING VISIONS INTO CODE

## Kollaborationstool

| Kriterium                           | ClickUp   | Notion   | Microsoft 365 Business   |
|-------------------------------------|---|--|--|
| <b>Beschreibung</b>                 |  <p>Ein umfassendes Projektmanagement-Tool mit Funktionen für Aufgabenverwaltung, Workflows, Automatisierungen und Team-Kollaboration.</p> |  <p>Eine flexible Plattform für Notizen, Wissensmanagement und Datenbanken, die sich für die Organisation und Bearbeitung von Informationen eignet.</p> |  <p>Eine umfassende Suite aus Produktivitäts- und Kollaborationstools, darunter Word, Excel, PowerPoint, Outlook, Teams und OneDrive, mit Cloud-Integration und KI-gestützten Funktionen.</p>   |
| <b>Plattform</b>                    | Web, Windows, Mac, Linux, Mobile  | Web, Windows, Mac, Mobile  | Web, Windows, Mac, Mobile  |
| <b>Echtzeit-Bearbeitung</b>         | Ja  | Ja   | Ja<br>(vor allem in Office-Dokumenten)   |
| <b>Dateifreigabe</b>                | Ja<br>(mit Cloud-Integration)   | Ja<br>(eigene Dateien + Verknüpfungen)   | Ja<br>(OneDrive, SharePoint)   |
| <b>Chat &amp; Messaging</b>         | Ja  | Nein (Kommentare & Notizen)  | Ja (über Microsoft Teams)  |
| <b>Videokonferenzen</b>             | Nein (nur durch Integrationen)  | Nein   | Ja   |
| <b>Projektmanagement</b>            | Ja (Kanban, Gantt, To-Do-Listen)  | Eingeschränkt (To-Do-Listen, Wikis)  | Eingeschränkt (Planner-Integration)  |
| <b>Versionierung</b>                | Ja (Änderungshistorie)  | Ja (Änderungshistorie)   | Ja (für Dokumente & Nachrichten)   |
| <b>Integrationen</b>                | Google Drive, Slack, Teams, Zapier  | Slack, Google Drive, Zapier  | Microsoft 365, Drittanbieter-Apps  |
| <b>Rechteverwaltung</b>             | Ja (granular einstellbar)   | Ja (Lesen, Bearbeiten, Admin)  | Ja (Teams, Kanäle, Benutzerrollen)   |
| <b>Datenschutz &amp; Sicherheit</b> | DSGVO-konform, Verschlüsselung  | DSGVO-konform, Verschlüsselung   | Hohe Sicherheit, DSGVO-konform   |
| <b>Benutzerfreundlichkeit</b>       | Mittel (Funktionsüberfluss)   | Einfach & intuitiv   | Mittel (bei vielen Funktionen)   |
| <b>max. Teamgröße</b>               | unbegrenzt  | Free: 10<br>Plus: 100<br>Business: 250   | 300  |
| <b>Ideal für</b>                    | Teams mit Fokus auf Projektmanagement   | Wissensmanagement, Dokumentation, persönliche Organisation   | Unternehmen mit Fokus auf Kommunikation & Zusammenarbeit   |
| <b>Kosten</b>                       | <p>Free Forever: Kostenlos</p> <p><b>Unlimited 6,52€ (7\$)</b> pro Benutzer/ pro Monat</p> <p><b>Business: 11,18€ (12\$)</b> pro Benutzer/ pro Monat</p>  | <p><b>Free: Kostenlos</b></p> <p><b>Plus: 9,50€</b> pro Benutzer/ pro Monat</p> <p><b>Business: 14€</b> pro Benutzer/ pro Monat</p>  | <p><b>1 Monat kostenlos testen</b></p> <p>verschiedene Abo-Pläne je nach Anforderungen</p> <p><b>Microsoft 365 Business Basic*:</b><br/>6,72 € pro Benutzer / pro Monat</p> <p><b>Microsoft 365 Business Standard*:</b><br/>14,04 € pro Benutzer / pro Monat</p> <p><b>Microsoft 365 Business Premium*:</b><br/>24,72 € pro Benutzer/ pro Monat</p> <p><b>Microsoft 365 Apps for Business*:</b><br/>11,76 € Benutzer/Monat</p> <p><b>*16% Ersparnis durch jährliche Abrechnung</b></p> |

### Auswahlbegründung:

Wir haben uns für **ClickUp** als Kollaborationstool entschieden, weil es ein hervorragendes Preis-Leistungs-Verhältnis bietet und bereits in der kostenlosen Version viele Funktionen enthält. Als All-in-One-App vereint es Aufgabenmanagement, Dokumente, Chats und Automatisierungen, wodurch zusätzliche Tools überflüssig werden. Zudem ist ClickUp besser für Teams geeignet als Notion, da es umfangreiche Funktionen für Workflows, Aufgabenverteilung und Zusammenarbeit bietet, was die Produktivität deutlich steigert.