

# Anwendung des Open-Source-Gebots im Projekt SmarterLeben

## 1. Einleitung

Im Projekt SmarterLeben, welches eines von 73 geförderten Modellprojekten Smart City (MPSC) in Deutschland ist, wird dem Open-Source-Gebot gemäß den [Vorgaben der Koordinierungs- und Transferstelle \(KTS\)](#) gefolgt. Dem Grundsatz Public Money – Public Code entsprechend müssen alle geförderten Entwicklungen unter einer Open Source Lizenz lizenziert werden, damit die entwickelten Lösungen übertrag- und nachnutzbar sind sowie Anbieter unabhängig betrieben werden können. Durch die Modellhaftigkeit einer erprobten Lösung wird die kommunale Handlungsfähigkeit gestärkt und die Kommunen zu mehr digitaler Souveränität befähigt.

Eine entwickelte Open Source Software Lösung (OSS) kann lizenzgebührenfrei vervielfältigt, bearbeitet oder weiterverbreitet werden. Damit dies gewährleistet ist, müssen auch die verwendeten Komponenten Open Source Lizenzen aufweisen und es muss auf freie Datenmodelle und Schnittstellen zurückgegriffen werden. Eine [Positiv- und Negativliste der Lizenzen wird auf OpenCoDE](#) bereitgestellt.

## 2. Dokumentation und Lizenzierung von Entwicklungen

Die Entwicklung wird im [KielRegion GitLab](#) dokumentiert und veröffentlicht, sodass sie im [Software-Verzeichnis von OpenCoDE](#) aufgenommen wird und damit für andere Kommunen nachnutzbar ist. Dabei sollen die [Nutzungsbedingungen von OpenCoDE](#) berücksichtigt werden. Mindestens folgende Dokumentations-Bausteine müssen vom Auftragnehmer erbracht werden:

Bestandteil	Beschreibung	Anforderung
README	<p>Erstellung einer README auf Deutsch und Englisch im Root-Verzeichnis mit folgenden Bestandteilen:</p> <ul style="list-style-type: none"> <li>• Einleitung, Beschreibung, Kontext</li> <li>• Erläuterung Funktionsumfang</li> <li>• <a href="#">Installations- und Betriebsdokumentation</a></li> <li>• <a href="#">Aktualisierung- und Entwicklerdokumentation (ggf. Contribution Guidelines)</a></li> <li>• <a href="#">Nutzerdokumentation</a></li> <li>• <a href="#">weitere Dokumentation</a></li> <li>• Code-Dokumentation (vgl. Kapitel 3: Code-Qualität)</li> <li>• Lizenzhinweis</li> </ul>	muss
publiccode.yml	Erstellung einer publiccode.yml für die <a href="#">Aufnahme ins Softwareverzeichnis</a>	muss
Rechteposition	Lizenzierung spätestens zum Zeitpunkt der Bereitstellung standardmäßig unter EUPL 1.2. Die Urheberrechte und einzuhaltenden Lizenzobligationen bereitgestellter Inhalte und derer Abhängigkeiten müssen durch technische Maßnahmen strukturiert erhoben und bereitgestellt werden. Abweichungen von der genannten Standard-Lizenz, z.B. im Zuge von Abhängigkeiten oder Inkompatibilitäten, müssen begründet werden. Der Lizenztext muss als LICENSE.txt im Root-Verzeichnis liegen.	muss
Software Bill of Materials	Erstellung einer Software Bill of Materials. Die SBOM bezeichnet die Auflistung der einzelnen Komponenten einer Software, bspw. im CyclonDX oder SPDX 2.2.1-Format.	muss

### 3. Richtlinien Code-Qualität

Um sicherzustellen, dass alle Beiträge von hoher Qualität sind und eine Nutzbarkeit der entwickelten Open Source Lösung gewährleistet wird, sind folgende allgemeine Richtlinien zu beachten. Die Umsetzung der Richtlinien soll im Angebot dargestellt werden:

Bestandteil	Beschreibung	Anforderung
Lesbarkeit	<p>Der Code sollte einfach zu lesen und zu verstehen sein. Dazu gehören:</p> <ul style="list-style-type: none"><li>• Verwenden von sprechenden Variablen- und Funktionsnamen</li><li>• Kommentare, die den Code erklären, ohne ihn zu wiederholen</li><li>• Einheitliche Code-Formatierung</li></ul>	soll
Struktur und Organisation	<p>Der Code sollte logisch strukturiert und organisiert sein:</p> <ul style="list-style-type: none"><li>• Funktionen und Methoden sollten eine klare und einzige Verantwortung haben</li><li>• Der Code sollte in logische Module oder Klassen unterteilt sein</li><li>• Es sollte eine klare Hierarchie von Abhängigkeiten geben</li></ul>	soll
Wiederverwendbarkeit	<p>Der Code sollte wiederverwendbar sein, um Redundanz zu vermeiden:</p> <ul style="list-style-type: none"><li>• Funktionen und Methoden sollten so geschrieben sein, dass sie in verschiedenen Kontexten verwendet werden können</li><li>• Der Code sollte auf allgemeine Probleme ausgerichtet sein, anstatt auf spezifische Anforderungen</li></ul>	soll
Fehlerbehandlung	<p>Der Code sollte robust gegenüber Fehlern sein:</p> <ul style="list-style-type: none"><li>• Fehler sollten explizit gehandhabt werden, anstatt ignoriert zu werden</li><li>• Der Code sollte so geschrieben sein, dass er sich von Fehlern erholen kann</li></ul>	soll

Testbarkeit	Der Code sollte einfach zu testen sein: <ul style="list-style-type: none"> <li>• Der Code sollte so geschrieben sein, dass er leicht getestet werden kann</li> <li>• Es sollten Unit- und Integration-Tests vorhanden sein, um den Code zu überprüfen</li> </ul>	soll
Konsistenz	Der Code sollte konsistent in Bezug auf Stil, Namensgebung und Struktur sein: <ul style="list-style-type: none"> <li>• Der Code sollte einem einheitlichen Stil und einer einheitlichen Namensgebung folgen</li> <li>• Der Code sollte so geschrieben sein, dass er leicht zu verstehen und zu warten ist</li> </ul>	soll
Dokumentation	Der Code sollte ausreichend dokumentiert sein: <ul style="list-style-type: none"> <li>• Der Code sollte Kommentare enthalten, die den Code erklären</li> <li>• Es sollten Dokumentationen vorhanden sein, die den Code und seine Funktionalität beschreiben</li> </ul>	soll

## 4. Umgang mit proprietärer Software

Baut die Entwicklung im Sinne einer Brown-Field Entwicklung auf bestehenden Strukturen oder Prozessen auf, so sind einige Ausnahmen von einer strengen Open Source Auslegung möglich.

Bei proprietärer Gerätesoftware gilt:

- Die digitalen Schnittstellen der Geräte müssen offen sein und einem Standard entsprechen, wenn in dem Anwendungsbereich bereits Standards existieren.
- Die höherwertige Software, die zur Vernetzung der Geräte verwendet wird, muss als Open-Source zur Verfügung stehen.
- Die Gerätesoftware selbst, die die Gerätedaten bis zur Geräteschnittstelle liefert, muss nicht als Open-Source zur Verfügung stehen.

Die Entwicklung von Schnittstellen zu proprietärer Software ist zulässig, wenn die Schnittstelle unter dem Open-Source-Gebot veröffentlicht wird.

Daten aus proprietären Systemen dürfen bei der Umsetzung von Maßnahmen genutzt werden (Finanzierung der Daten nicht durch Projektgelder).

Im Zusammenhang mit Projekten, die eine hohe Modellhaftigkeit haben bzw. die Wissensbasis der Kommunen über den bestehenden Stand erweitern, könnte im Einzelfall (nach Abstimmung mit der KTS) eine Förderung erfolgen.

Bei hochkomplexer Spezialsoftware für Fachanwendungen der Stadtentwicklung, wie GIS-Anwendungen, CAD sowie der Architektur von urbanen Datenplattformen, sind Schnittstellen zu proprietären Systemen regelmäßig förderfähig, wenn diese Schnittstellen als Open Source zur Verfügung gestellt werden. Insbesondere für Tools/Anwendungen aus dem Bereich Geoinformationen existieren interoperable Standards und weit verbreitete Open-Source-Alternativen. Die Notwendigkeit der Verwendung von proprietären Schnittstellen ist hier gut zu belegen (Vertragsbedingungen, bzw. Nischenprodukte). Diese Fälle unterliegen einer Einzelprüfung.

## **5. Offene Daten (Open Data)**

Werden Daten im Rahmen einer Entwicklung als offene Daten bereitgestellt, so gilt die Berücksichtigung der Richtlinie „Offene Daten und die Wiederverwendung von Informationen des öffentlichen Sektors“, Richtlinie (EU) 2019/1024 (hat die PSI-Richtlinien abgelöst):

- Richtlinie basiert auf dem allgemeinen Grundsatz, dass öffentliche und öffentlich finanzierte Daten für kommerzielle oder nichtkommerzielle Zwecke weiterverwendbar sein sollten.
- Die Weiterverwendung von Dokumenten steht allen Marktteilnehmern offen und alle geltenden Weiterverwendungsbedingungen sollten nicht-diskriminierend sein.
- Vermeidung oder Einschränkung einzigartiger nationaler (oder regionaler) Lizenzen
- Standardmäßig sind CC-Lizenzen, insbesondere [CC-BY](#) und CC0
- Vermeidung von Share-Alike-Lizenzen und Lizenzen, die auf die nichtkommerzielle Nutzung oder die Nutzung innerhalb eines bestimmten Sektors oder Anwendungsfalls beschränkt sind