Plugin-Entwicklung: Initiale / Kollaborative kontinuierliche Entwicklung

Dies ist ein Auszug aus dem Leitfaden 'Plugin-Entwicklung' der Plugin-Werkstatt von Moodle an Hochschulen e.V., der nach und nach in die Knowledge Base übertragen wird.

- Plugin-Entwicklung: Zielsystem
- Plugin-Entwicklung: Entwicklungsumgebung
- Plugin-Entwicklung: Orte des Wissens zu Coding-Standards und APIs
- Plugin-Entwicklung: Übersichten zu Standards
- Plugin-Entwicklung: Form des Entwicklungsprozesses
- Plugin-Entwicklung: Zeitaufwand
- Plugin-Entwicklung: Kompatibilität Moodle-Versionen
- Plugin-Entwicklung: Testen
- Plugin-Entwicklung: Maintenance
- Plugin-Entwicklung: Best practices Dokumentation

Plugin-Entwicklung: Zielsystem

Auf welchen Zielsystemen (z.B. Moodle-Version, PHP-Version) soll das Plugin laufen?

Das Moodle HQ pflegt eine Übersicht über vergangene und zukünftige Releases. Wenn ein Plugin neu entwickelt wird, ist es lohnenswert, für alle aktuell unterstützen Moodle Versionen zu entwickeln. Dabei sollte auch ein Augenmerk auf die LTS Releases gelegt werden (deren Anforderungen möglicherweise bereits veraltete PHP Versionen beinhalten). Die Moodle Versionen wiederum stellen Anforderungen an PHP und Datenbanken. Das Plugin sollte auf jeden Fall MySQL/MariaDB sowie Postgres in der entsprechenden Version unterstützen.

Die meisten Moodle-Installationen laufen auf einer Linux-Variante. Es ist daher empfohlen ein Linux zu verwenden, wo der Code getestet wird. Gleichwohl wird auch Windows offiziell unterstützt. Entwickler, die auf Windows entwickeln, können Moodle in Docker-Containern oder in einer VM mit Linux laufen lassen.

Autor: Stephan Robotta, Berner Fachhochschule

Plugin-Entwicklung: Entwicklungsumgebung

Wo sollte ich mein Plugin entwickeln (lokal, Testserver, ..)?

Um ein Moodle-Plugin zu entwickeln, ist es am einfachsten eine Entwicklungsumgebung für Moodle (also eine lokale Moodle-Installation) auf dem eigenen Rechner einzurichten.

Hierzu gibt es 2 grundsätzliche Wege:

- Moodle MDK (nur MacOS und Linux)
- Docker (Windows mit WSL, MacOS und Linux)

Vom Moodle HQ werden beide Wege mit entsprechenden Repositories unterstützt. Welchen Weg man davon geht, hängt von den eigenen Vorlieben und Erfahrungen sowie dem eigenen Betriebssystem ab.

Zum Testen werden nach der Installation des Moodles in der Entwicklungsumgebung am besten ein Kurs sowie zwei Nutzerkonten angelegt. Ein User bekommt die Rolle „Teacher“, ein anderer die Rolle „Student“. Der Admin-Account ist durch die Installation bereits vorhanden.

Autor: Stephan Robotta, Berner Fachhochschule

Plugin-Entwicklung: Orte des Wissens zu Coding-Standards und APIs

Wo finde ich Infos zu Coding-Standards und APIs?

Erste Anlaufadresse für Coding-Standards und Dokumentation ist moodledev.io. Gleichzeitig ist es sinnvoll, in bestehenden Code in Moodle selbst oder auch in Plugins zu schauen, um eventuell eine Lösung für das eigene Problem zu finden.

Auf der Seite Support Channels sind einige Möglichkeiten (Forum, Matrix Chat) gelistet, wie man mit anderen Moodle-Entwicklern in Kontakt treten kann, um Fragen zu stellen und Erfahrungen auszutauschen.

Moodle bietet eine Plugin CI an, die für die Continuous Integration eines Plugins genutzt werden kann. Darin sind Checks und Tests enthalten, die den Code auf Konsistenz und Regeln prüfen. Für Plugin-Entwickler ist es nicht zwingend vorgeschrieben, sich an diese Regeln zu halten. Es wird aber empfohlen den Regeln zu folgen, um für Außenstehende eine nachvollziehbare Qualität des Plugins zu liefern und zu demonstrieren.

Autor: Stephan Robotta, Berner Fachhochschule

Plugin-Entwicklung: Übersichten zu Standards

Welche Standards muss ich beachten (z.B. Coding Guidelines, Datenschutz, Barrierefreiheit)?

Überblick **Moodle Policies**

Informationen zu den **Moodle Coding Guidelines**

Ein einheitlicher Programmierstil ist in jedem Entwicklungsprojekt wichtig, insbesondere wenn viele Entwickler beteiligt sind. Ein standardisierter Stil trägt dazu bei, dass der Code leichter lesbar und verständlich ist, was die Gesamtqualität verbessert.

Informationen zu den **Moodle Accessibility Guidelines**

Grundsätzlich gilt, dass alle Komponenten von Moodle für alle Benutzer verfügbar sein müssen. Barrierefreiheit muss bei der Entwicklung jeder neuen Funktion in Moodle berücksichtigt werden. Moodle strebt die Umsetzung von WCAG 2.2 Level AA an. Die Einhaltung dieses Standards wird in regelmäßigen Audits extern anhand ausgewählter Beispielseiten zertifiziert. In Deutschland ist allerdings die strengere BITV 2.0 maßgeblich.

Informationen zur **Moodle Privacy API**

Die Privacy API ermöglicht es, die von Moodle gespeicherten personenbezogenen Daten zu beschreiben und bietet die Möglichkeit, diese Daten nutzerbezogen zu ermitteln, zu exportieren und zu löschen. Dies gewährleistet die Einhaltung von Vorschriften wie der Datenschutz-Grundverordnung (DSGVO) in der EU.

Wenn ein Plugin im Moodle Marketplace veröffentlicht werden soll, müssen diese Standards eingehalten werden. Das ist in den Regeln zur Plugin Contribution ausformuliert.

Die Ziele einer „sauberen“ Programmierung im Sinne dieser Standards sind:

- Reduzierung der Fehleranfälligkeit
- Bessere Wartbarkeit

- Bessere Weiterentwickelbarkeit

Autorin: Melanie Treitinger, Ruhr-Universität Bochum

Plugin-Entwicklung: Form des Entwicklungsprozesses

Wie sollte der Entwicklungsprozess aussehen (Top-Down vs. Agil)?

Wenn ein Plugin neu entwickelt wird, gibt es in der Regel eine Anforderung, welche mit aktuellen Mitteln nicht erfüllt werden kann. Es ist empfohlen, diese Anforderung in User Stories zu formulieren, um dann einzelne Tasks bzw. Projekt-Teile aus den User Stories zu erstellen. [Siehe auch Plugin-Entwicklung: Anforderungssammlung]

Beim Start ist es unwichtig, ob eine agile oder eine Top-Down-Strategie zum Entwickeln angewendet wird. Die agile Methode ist sicher moderner und lässt sich u.U. mit gängigen Projektmanagement-Tools besser abbilden.

Es ist sinnvoll, den Entwicklungsprozess in einzelne Teilprojekte mit Tasks zu zerlegen und diese auch entsprechend, z.B. in Github-Issues, auszuformulieren. Auf diese Weise ist der Prozess auch für Außenstehende nachvollziehbar und man kann bspw. Entwicklerkapazität auch aus der Community beziehen. [Siehe auch Plugin-Entwicklung: Anforderungsdokumentation]

Autor: Stephan Robotta, Berner Fachhochschule

Plugin-Entwicklung: Zeitaufwand

Wie viel Zeit benötige ich zum Entwickeln?

Die Antwort auf diese Frage hängt von folgenden Faktoren ab:

- Umfang und Komplexität der Anforderungen
- Umfang und Detailliertheit des Lastenhefts (User Stories, Mockups)
- Erfahrungsstand der entwickelnden Person(en) mit Moodle

Grundsätzlich sollte man bei der Entwicklung von Software immer einen zeitlichen Puffer einbauen, empfohlen werden meist 20%.

Der Zeitrahmen für die Fertigstellung hängt dementsprechend von diesen Faktoren ab.

Autorin: Melanie Treitinger, Ruhr-Universität Bochum

Plugin-Entwicklung: Kompatibilität Moodle-Versionen

Wie kann ich die Kompatibilität mit unterschiedlichen Moodle-Versionen sicherstellen (Moodle Plugin-CI)?

Unterschiedliche Moodle-Versionen bedeuten in der Regel auch unterschiedliche PHP-Versionen sowie Datenbank-Versionen, für die das Plugin funktionsfähig sein sollte.

Um die Kompatibilität sicherzustellen, bieten sich automatisierte Tests an. Moodle stellt dafür eine eigene Konfigurationsdatei für Github Actions bereit. Hierzu existiert auch eine passende Anleitung zur Nutzung von Github Actions.

Autorin: Melanie Treitinger, Ruhr-Universität Bochum

Plugin-Entwicklung: Testen

Wie teste ich mein Plugin?

Bei der Entwicklung sollte man PHPUnit-Tests und Behat-Tests als automatisierte Tests mit implementieren.

Daneben sind manuelle Tests nach vorher definierten Szenarien („wenn ich Einstellung X aktiviere sollten Lehrende Funktion Y nutzen können“) sinnvoll.

Autorin: Melanie Treitinger, Ruhr-Universität Bochum

Plugin-Entwicklung: Maintenance

Wie kann ich eine Maintenance sicherstellen?

Ein Plugin sollte mindestens nach jeder Veröffentlichung einer neuen Moodle Major-Version auf Kompatibilität geprüft und angepasst werden.

Je nach Umfang und Komplexität des Plugins sollte ein entsprechendes Budget dafür vorgesehen werden. Der Dienstleister kann dafür auch ein Angebot machen.

Autorin: Melanie Treitinger, Ruhr-Universität Bochum

Plugin-Entwicklung: Best practices Dokumentation

Was sind Best Practices für Nutzer*innen- und Entwickler*innendokumentation?

Nutzer*innendokumentation (User Documentation)

Hier steht die Problembehebung und Anwendung im Vordergrund. Nutzer*innen wollen wissen, wie sie mit der Software ein bestimmtes Ziel erreichen. Der Aufbau kann entweder rollenbasiert oder aufgabenbasiert oder eine Kombination aus beidem sein.

Best Practice Beispiel: Niels Gandraß – Archiving Subsystem for Moodle (Usage)

Entwickler*innendokumentation (Developer Documentation)

Hier steht die Erweiterbarkeit, Integration und Wartung im Vordergrund.

Die Systemarchitektur, das zugrundeliegende Datenmodell sowie eventuelle Abhängigkeiten sollten erklärt werden. APIs sollten nach anerkannten Standards dokumentiert werden (z.B. OpenAPI). Code-Beispiele erleichtern die Erweiterbarkeit.

Best Practice Beispiel: Niels Gandraß – Archiving Subsystem for Moodle (Developer Documentation)

Autorin: Melanie Treitinger, Ruhr-Universität Bochum