hochschule burgenland
University of Applied Sciences

PRAXISPROJEKTE 2026

BACHELOR
SOFTWARE ENGINEERING UND VERNETZTE SYSTEME



WIR BRINGEN BESONDERES ZUSAMMEN.

hochschule-burgenland.at

Fachhochschul-Bachelorstudiengang

Software Engineering und vernetzte Systeme



*„Egal, in welchen Bereich des Lebens man blickt, Software ist ein wesentlicher Bestandteil jeder Innovation. Unsere Absolvent*innen sind so der Schlüssel zum Erfolg neuer Lösungen.“*

Franz Knipp, Studiengangsleiter

Das Bachelorstudium verfolgt das Ziel, Software-Entwickler*innen auszubilden, die den Entwicklungsprozess von der initialen Idee bis zur Inbetriebnahme verstehen und auf Basis aktueller Technologien umsetzen können. Neben der Vermittlung des theoretischen Wissens bildet die praktische Umsetzung einen wichtigen Bestandteil der Ausbildung.

„Softwareentwicklung ist mehr als Programmieren. Organisations- und Kommunikationskompetenzen machen ein Projekt erst erfolgreich.“

Christian Büll, Departmentleiter



Im vierten Semester beweisen die Studierenden die erworbenen Kompetenzen im Rahmen eines mehrmonatigen Projekts. Neben den technischen Fertigkeiten der Softwareentwicklung und -architektur sind die Teams auch für das Management des Projekts und die Kommunikation mit der Auftraggeber*in zuständig.

Dieses Projekt wird übergreifend in den Lehrveranstaltungen Praxisprojekt, Softwarequalität und Projektmanagement II umgesetzt.

Inhalt

1 Praxisprojekte **2**
1.1 Weiterentwicklung gAmIcode 2
1.2 CourseGuide 6
1.3 KNX Virtual 2.0 - User Interface Design 9
1.4 Gamified Evidence-Based Nursing (GEBN) 12
1.5 Pegelhub – Connectoren und Funktionsmodule v6 15

2 Projektmitglieder – BSWE Jahrgang 2024 **18**

3 Projektbetreuer*innen **20**

4 Verzeichnisse **21**

5 Impressum **23**

gAmIcode

Weiterentwicklung zu einer modularen, erweiterbaren Architektur mit automatisierter Code-Bewertung



Projektauftraggeber*in: Sebastian Saip
Projektbetreuer*in: Dipl.-Ing.in Patrizia Sailer, BSc MSc MBA
Projektmitglieder: Aurika Afanaseva, Nicolas Haselbacher, Jeremy Heißenberger, Marco Magyar
Repository: https://github.com/fhburgenland-bswe/gamification

gAmIcode ist die gamifizierte Lernplattform der Hochschule Burgenland: Studierende bearbeiten Programmieraufgaben im Browser, erhalten automatisiertes Feedback und sammeln Erfahrungspunkte, Level und Badges. Im Praxisprojekt wurde die Code-Ausführung des bestehenden Meteor-Systems in eine modulare, containerbasierte Architektur überführt. Ein zentrales Gateway leitet Einreichungen an isolierte Runner für Java, SQL und HTML/CSS weiter; eine einheitliche Schnittstelle trennt Editor, Ausführung und Bewertung. Das Team erweiterte außerdem das MongoDB-Datenmodell, migrierte die Live-Vorschau von StackBlitz WebContainer zu Nodepod und ergänzte eine Lektorenansicht für die Aufgabenverwaltung.

Auftraggeber

Auftraggeber des Projekts ist Sebastian Saip, Lektor an der Hochschule Burgenland. Er betreibt gAmIcode im Lehrbetrieb und setzt die Plattform in Lehrveranstaltungen der Softwareentwicklung ein. Studierende gelangen dabei typischerweise über Learning Tools Interoperability (LTI) aus Moodle in die Lernumgebung.

Patrizia Sailer betreute das Projekt als Product Ownerin, übernahm die fachliche Priorisierung und prüfte die Ergebnisse gegen die Projektvision. Sebastian Saip betreute das Projekt technisch, stellte die Server-Infrastruktur bereit und nahm die Ergebnisse in den Sprint-Reviews ab. Ausgangspunkt des Projektauftrags waren zwei Probleme im Lehrbetrieb: Neue Kurse und Aufgaben ließen sich in der bestehenden Struktur nur mit hohem Aufwand anlegen, und jede zusätzliche Programmiersprache erforderte tiefe Eingriffe in den Plattformkern.

Projektbeschreibung

gAmIcode basiert auf Meteor.js, React und MongoDB und ist über LTI 1.3 in Moodle eingebunden. Zu Projektbeginn lief die HTML/CSS-Bewertung direkt im Meteor-Backend; die bereits im Browser ausgeführte Live-Vorschau war an StackBlitz WebContainer gebunden. Der gewachsene Anwendungskern sollte erhalten bleiben, während die Ausführungslogik gezielt modularisiert wurde.

Vorgesehen waren persistente, technologiespezifische Runner-Container und eine einheitliche, an-

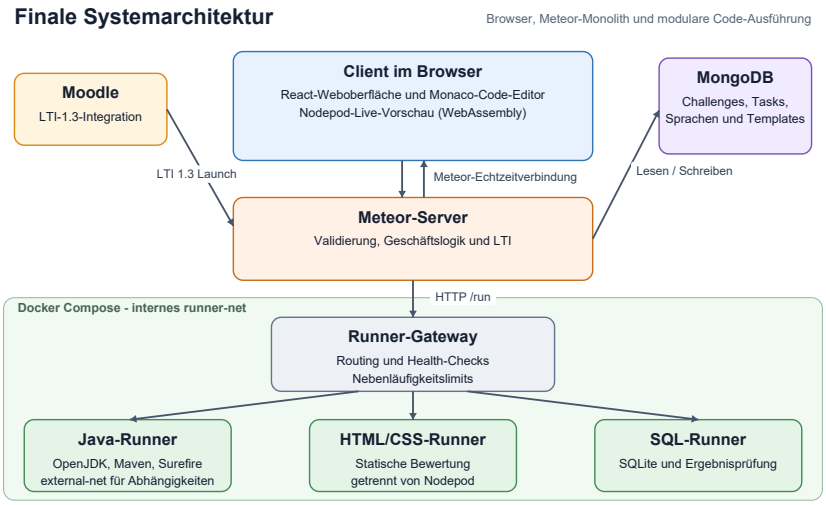


Abbildung 1: *Komponenten und Kommunikationswege der finalen gAmIcode-Systemarchitektur*

fragebasierte Schnittstelle zwischen Code-Editor und Ausführungsumgebungen. Die Bewertung sollte je Technologie Testergebnisse, Ausgaben, Kompilierfehler und Zeitüberschreitungen zurückgeben. Zusätzlich waren ein generisches Datenmodell sowie eine abgesicherte Ausführung in isolierten, gehärteten Containern mit Timeouts, Ressourcenlimits und Nebenläufigkeitssteuerung gefordert. Mindestens Java oder SQL sollte als Proof of Concept (PoC) die neue Architektur belegen; die bestehende Funktionalität musste erhalten bleiben.

Umgesetzt wurden ein Java-Runner mit Maven, ein SQL-Runner und ein eigener HTML/CSS-Runner, in den die bestehende Bewertung überführt wurde. Ein Runner-Gateway übernimmt Routing und Health-Checks, während die getrennten Docker-Container die technologiespezifische Ausführung kapseln. Das MongoDB-Modell wurde dafür um generische Technologie-, Editor- und Testkonfigurationen erweitert. Beide optionalen Ziele wurden erreicht: SQL kam als zweite Technologie hinzu, und die Lektorenansicht wurde umgesetzt.

Die gemeinsame Schnittstelle vereinheitlicht den technischen Ablauf, ohne die Unterschiede der Technologien zu verbergen. Der Editor übergibt Quellcode, Test- und Laufzeitkonfigurationen an das Gateway; dieses wählt den passenden Runner aus und prüft dessen Verfügbarkeit. Jeder Runner kapselt die konkrete Ausführung und liefert Ergebnisse, Ausgaben sowie Fehler in einer einheitlichen Antwortstruktur zurück. Dadurch bleiben Routing und Darstellung im Meteor-System technologieunabhängig, während Maven-, SQL- und HTML/CSS-spezifische Logik in den jeweiligen Containern liegt.

Im Projektverlauf wurde der Umfang um zwei priorisierte Änderungen erweitert: Die HTML/CSS-Bewertung wurde in die gemeinsame Runner-Struktur überführt, und die Live-Vorschau wechselte von StackBlitz zu Nodepod. Beide Themen sind voneinander getrennt: Die statische Bewertung läuft serverseitig im Runner, die Vorschau weiterhin im Browser. Nicht Teil des Projekts waren neue Gamification-Funktionen, ein Redesign des Levelsystems, ein Umbau des Editors oder qualitatives Feedback zum Programmierstil.

Projektverlauf

Das Projekt startete am 6. März 2026 und wurde in fünf Sprints nach Scrum-orientierter Arbeitsweise

mit Jira, Xray und Confluence umgesetzt. Präsentation und Übergabe folgten am 26. Juni. Im ersten Sprint (6.–27. März) arbeitete sich das Team in die gewachsene, stellenweise knapp dokumentierte Codebasis ein und erarbeitete das Architektur- und Schnittstellenkonzept, das termingerecht stand. MongoDB war für alle neu; Pair-Programming und laufende Dokumentation halfen beim Einstieg.

Im zweiten Sprint (27. März–17. April) entstanden die Abstraktionsebene, die Kernlogik der Code-Ausführung und die Erweiterungen am Datenmodell. Im dritten Sprint (17.–30. April) wuchs der Projektumfang deutlich: Zur laufenden Java-Integration kam die Auslagerung der HTML/CSS-Bewertung in einen eigenen Runner hinzu, und zusätzlich wurde – hoch priorisiert – die Ablöse des externen WebContainer-Anbieters StackBlitz als neue Anforderung aufgenommen. Die Umsetzung des Java-Runners mit Maven einschließlich der Auswertung von Surefire-Berichten erforderte 72,5 statt der geplanten 40 Stunden und wurde erst im vierten Sprint fertiggestellt. Das ursprünglich eigene Arbeitspaket zur Evaluierungslogik ging dabei in den Runner-Arbeiten auf.

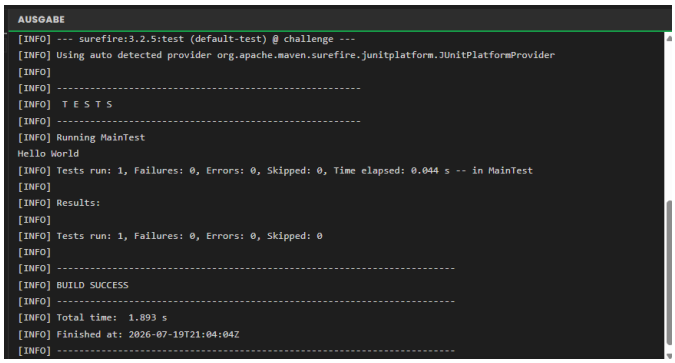


Abbildung 2: *Maven-Testausgabe des Java-Runners im Code-Editor*

Im vierten Sprint (30. April–15. Mai) ergänzte das Team für den Java-Runner Timeouts sowie Ressourcen- und Nebenläufigkeitslimits. Parallel wurde die Ablöse von StackBlitz analysiert und für den fünften Sprint vorbereitet. Einzelne User Stories waren um nicht abgestimmte Ideen erweitert worden und mussten auf den vereinbarten Umfang zurückgeführt werden. Die Retrospektive führte deshalb zur Regel, keine ungefragten Features umzusetzen.

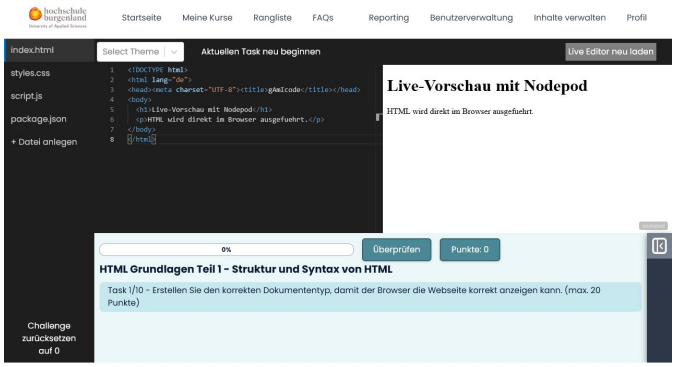


Abbildung 3: *Live-Vorschau einer HTML-Aufgabe mit Nodepod*

Im fünften Sprint (15.–29. Mai) wurde die Live-Vorschau von StackBlitz WebContainer auf Nodepod umgestellt. Dafür waren drei Anpassungen am Service Worker und zusätzliche Browser-Isolationsheader für WebAssembly-Threading nötig. Parallel entstand der SQL-Runner als zweite Technologie; dafür wurden 50 Stunden erfasst, gegenüber 72,5 Stunden für den Java-Runner. Wegen der technischen Unterschiede lässt sich daraus jedoch keine belastbare Aussage über den allgemeinen Erweiterungsaufwand ableiten.

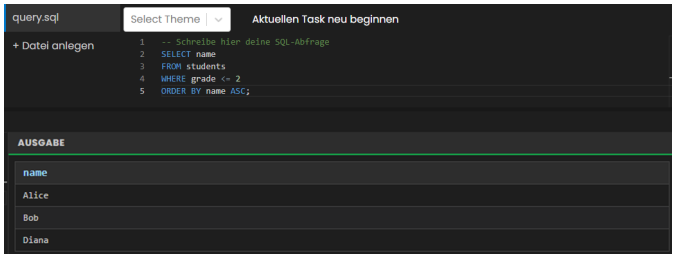


Abbildung 4: *SQL-Abfrage mit Ergebnistabelle des SQL-Runners*

Für die optionale Lektorenansicht wurden 99,5 statt der geplanten 40 Stunden benötigt; damit

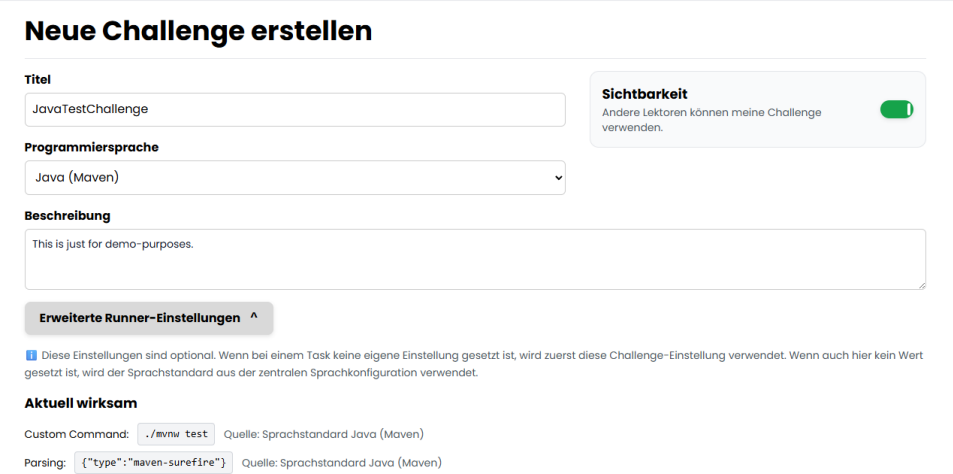


Abbildung 5: *Lektorenansicht mit wirksamen Maven- und Parsing-Einstellungen*

war sie der größte einzelne Implementierungsposten. Lehrende verwalten damit Start- und Testdateien sowie Runner-Einstellungen über die Oberfläche. Beim ersten Speichern neuer Java/Maven- und SQL-Tasks werden Standard-Testdateien erzeugt. Für Custom Command und Parsing gilt die Fallback-Kette Task → Challenge → Sprachstandard.

Die Qualitätssicherung lief sprintbegleitend und risikoorientiert in Anlehnung an ISO 29119-3. Xray-Testfälle waren mit User Stories verknüpft; Test Plans begleiteten die Sprints. Gemerged wurde erst nach Code-Review und grüner CI-Pipeline mit Commit- und Quellcodeprüfungen sowie Unit-, Integrations- und E2E-Tests. Am Projektende standen 145 Xray-Testfälle und 219 Pipeline-Läufe; die erfolgreichen Läufe dauerten im Mittel rund 56 Minuten.

Für Fehleranalyse und Bug-Fixing wurden 76 von 200 geplanten Stunden benötigt. Alle sieben Meilensteine wurden termingerecht erreicht; 1.185,5 Ist-Stunden standen 1.485 adaptierten Planstunden gegenüber. Der Aufwand war weitgehend gleichmäßig auf die vier Teammitglieder verteilt.

Fazit

Zum Projektabschluss liefen Java, SQL und HTML/CSS über dieselbe Runner-Architektur; Meteor, React und MongoDB blieben der Kern der Plattform. Der SQL-Runner belegt, dass sich eine weitere Technologie über die gemeinsame Schnittstelle ergänzen lässt; jeder Runner benötigt dennoch eine eigene Ausführungs- und Bewertungslogik.

Die Einarbeitung in die bestehende Codebasis und die Änderungen des Projektumfangs prägten den Projektverlauf stärker als eine einzelne Technologie. Die Lektorenansicht zeigte mit 99,5 statt 40 Stunden, wie schwer sich oberflächennahe Funktionen mit vielen fachlichen Details im Voraus schätzen lassen. Bewährt haben sich klare Rollen, frühe fachliche Abstimmung, Code-Reviews und laufende Dokumentation.

Welche Technologie als Nächstes integriert wird, sollte sich aus einem konkreten Anwendungsfall in der Lehre ergeben. Auch diese Anbindung muss mit passenden Sicherheits-, Bewertungs- und Regressionstests abgesichert werden.

Autor*innen: Aurika Afanaseva, Nicolas Haselbacher, Jeremy Nicolas Heißenberger, Marco Magyar

CourseGuide

Courseguide.

Projektauftraggeber*in: Ing. Christian Putzenlechner, BA BEd MA (Leiter, Center Innovative Lehre)
Projektbetreuer*in: Dipl.-Ing. Franz Knipp
Projektmitglieder: Alexander Brenner, Süleyman Flachberger, Alexander Grill, Thomas Michalek
Repository: https://gitlab.com/bswe-public/courseguide

CourseGuide ist eine Moodle-Erweiterung, die Lehrende dabei unterstützt, verpflichtende Kursinformationen vollständig, korrekt und nachvollziehbar zu pflegen. Eine direkt im Moodle-Kurs integrierte Ampelanzeige zeigt auf einen Blick, welche Informationen vorhanden sind und wo noch Handlungsbedarf besteht. Das Projekt entstand in enger Zusammenarbeit mit dem Center Innovative Lehre der Hochschule Burgenland und wurde iterativ über fünf Sprints entwickelt. Neben der Softwarelösung entstanden eine containerisierte Entwicklungsumgebung, eine automatisierte Teststrategie mit insgesamt 103 Testfällen, eine vollständige und umfassende Projektdokumentation sowie die Vorbereitung eines ersten geplanten Pilotbetriebs für das produktive Moodle-System der Hochschule Burgenland im Herbst 2026.

Auftraggeber

Auftraggeber des Projekts war das Center Innovative Lehre der Hochschule Burgenland. Das Center begleitet Lehrende bei der Gestaltung digitaler Lehr- und Lernprozesse und betreut die zentrale Moodle-Plattform der Hochschule.

Für jede Lehrveranstaltung müssen zahlreiche verpflichtende Informationen gepflegt werden: Kursmetadaten, Semesterwochenstunden, ECTS, Kompetenzerwerb, Lehrinhalte, Beurteilungskriterien, Literatur sowie Kontaktdaten der Lehrenden. In der Praxis sind diese Angaben jedoch häufig unvollständig oder uneinheitlich – besonders bei neu angelegten Kursen oder bei externen Lehrenden. Die Nachkontrolle erfolgt dadurch oft manuell und verursacht zusätzlichen organisatorischen Aufwand.

CourseGuide setzt genau hier an: Die Erweiterung soll Lehrende nicht kontrollieren oder bewerten, sondern als „Wegweiser“ durch den gesamten Prozess der Kurserstellung unterstützen. Eine direkt im Kurs sichtbare Ampelanzeige liefert eine verständliche Rückmeldung über den Vollständigkeitsgrad, sodass fehlende Angaben frühzeitig ergänzt werden können. Damit wird Unterstützung direkt im gewohnten Moodle-Arbeitsablauf möglich.

Projektbeschreibung

Ziel des Projekts war eine Moodle-Erweiterung mit übersichtlichem Ampelsystem. Beim Öffnen eines Kurses zeigt ein Badge den Gesamtstatus in Grün, Gelb oder Rot. Über die Detailansicht lassen sich Prüfergebnisse je Bereich nachvollziehen und gezielt nachbessern. Die Bewertung erfolgt automatisch beim Laden des Kurses, ohne zusätzliche Schritte für Lehrende. Lehrende erhalten so jederzeit Orientierung, ohne den Kurs verlassen zu müssen.



Abbildung 6: *Das Ampelsystem in Grün, Gelb und Rot*

CourseGuide prüft Kursmetadaten, Kompetenzerwerb, Lehrinhalte, Beurteilungskriterien, Kontaktinformationen und Literaturangaben – nicht die didaktische Qualität, sondern ausschließlich *Vollständigkeit*, *Struktur* und *formale Korrektheit*. Die Literaturliste wird dabei als eigene Warnkategorie behandelt.

Besonders anspruchsvoll gestaltete sich die Validierung der Beurteilungstabelle. Geprüft wurden Struktur, Sichtbarkeit und Gewichtung der Teilbeurteilungen. Da Moodle unterschiedliche Eingabeformen erlaubt, kamen reguläre Ausdrücke und regelbasierte Heuristiken zum Einsatz. Künstliche Intelligenz wurde bewusst nicht eingesetzt. Alle Prüfungen bleiben reproduzierbar, testbar und für Folgeprojekte übernehmbar. Eine unsichtbare Literaturliste führt beispielsweise zu einem Gelb-Status und zeigt Handlungsbedarf an.

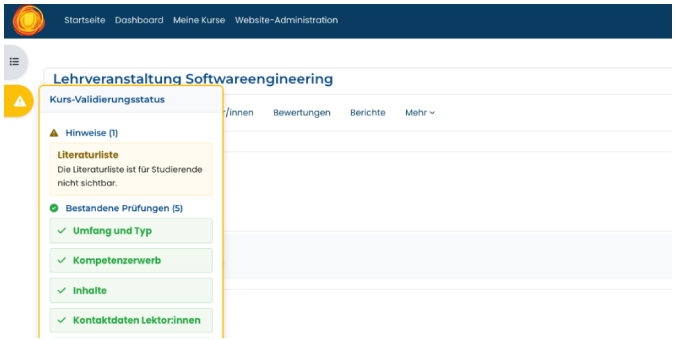


Abbildung 7: *Warnstatus bei nicht sichtbarer Literaturliste*

Technisch wurde *CourseGuide* als lokale Moodle-Erweiterung umgesetzt. Das Plugin greift nur lesend auf bestehende Moodle-Strukturen zu und verzichtet auf eigene Datenbanktabellen oder zusätzliche Schnittstellen (API). Validierungslogik, UI und Darstellung wurden getrennt: Services für die Prüfung, Renderer und Mustache-Templates für die Anzeige. Daraus entstand eine Teststrategie mit 103 Unit- und E2E-Tests, die per Integration automatisch mit Xray-Testfällen verknüpft werden.

Projektverlauf

Zu Beginn stand weniger die technische Umsetzung als das Verständnis der Moodle-Umgebung und die Klärung verpflichtender Kursinformationen mit dem Auftraggeber im Vordergrund. Das Projekt wurde in fünf Sprints umgesetzt, mit wöchentlichen Abstimmungen, Reviews mit Product Owner und Auftraggeber sowie zwei Retrospektiven. Anforderungen und Fehler wurden in Jira strukturiert. Die Jira-GitLab-Integration verknüpfte Branches und Merge Requests automatisch mit den Tickets.

Früh entstand ein Prototyp des Ampel-Badges und wurde in einem Moodle-Kurs angezeigt. Die fachliche Validierung erwies sich als schwieriger, weil einfache Zeichenkettenvergleiche in Moodle nicht ausreichten. Aus dem Prototyp entstanden nach und nach die fachlichen Validatoren.

Da keine Moodle-Instanz die Hochschulkonfiguration abbildete, entwickelte das Team eine containerisierte Entwicklungsumgebung mit Plugins, Themes und hochschulspezifischen Einstellungen. Parallel sicherte das Team die Softwarequalität ab: 356 Work Items in Jira, verpflichtende Code Reviews und 103 automatisierte Tests mit Dokumentation über die CI/CD-Pipeline in Xray. Die lokale Umgebung bildete dieselbe Konfiguration ab wie die Pipeline und erleichterte so reproduzierbare Tests vor jedem Merge.

Alexander Brenner verantwortete Projektleitung, Architektur, Infrastruktur sowie Test- und Deploymentstrategie. Süleyman Flachberger und Alexander Grill entwickelten die fachliche Validierungslogik, Thomas Michalek wirkte schwerpunktmäßig an der Testautomatisierung mit.

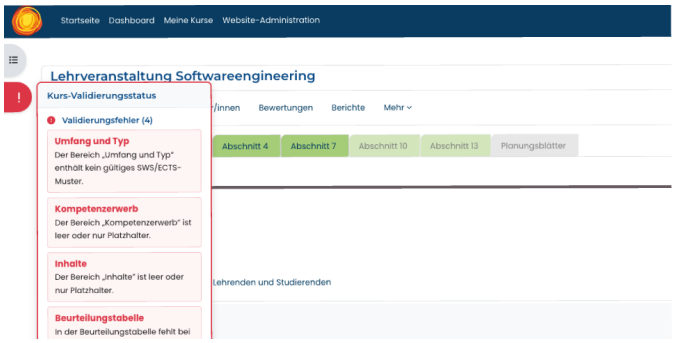


Abbildung 8: *Detailansicht mit Validierungsfehlern*

Jeder Validierungsfehler wird mit einer verständlichen Meldung und einem Verweis auf den betroffenen Kursbereich angezeigt. Die enge Zusammenarbeit aller Teammitglieder erwies sich dabei als wesentlicher Erfolgsfaktor. Alle Ergebnisse und Entscheidungen wurden fortlaufend in Confluence dokumentiert und für die spätere Übergabe aufbereitet. Damit wurde eine Erweiterung geschaffen, die im Alltag der Lehrenden unmittelbar nutzbar und für Folgeprojekte wartbar bleibt.

Fazit

CourseGuide vereint als Moodle-Erweiterung fachliche Anforderungen, Softwarequalität und strukturiertes Projektmanagement in einer praxistauglichen Lösung. Die größte Schwierigkeit lag nicht in der UI, sondern in der Validierung anhand der internen Moodle-Strukturen, wo selbst einfache Prüfregeln nur mit erheblichem Testaufwand abgesichert werden konnten.

Besonders wertvoll war der kontinuierliche Austausch mit dem Center Innovative Lehre und den Lehrenden, der die Erweiterung konsequent an den tatsächlichen Bedürfnissen ausrichtete. Übergeben wurden außerdem Architekturbeschreibung, Validierungsregeln, technische Dokumentation und ein Benutzerhandbuch für Lektor*innen. Zum Projektabschluss wurde ein Release Candidate für Moodle 4.1 vorbereitet. Der Pilotbetrieb im produktiven System ist für Herbst 2026 vorgesehen.



Abbildung 9: *Vollständig validierter Kurs*

Für die Hochschule entfällt damit ein Teil der bisherigen manuellen Nachkontrolle bei der Kurseröffnung. Der Ansatz bleibt bewusst schlank, erweiterbar und an bestehende Moodle-Standards angelehnt. Damit bildet *CourseGuide* eine stabile Grundlage für den weiteren produktiven Einsatz an der Hochschule Burgenland.

Autor*innen: Alexander Brenner, Süleyman Flachberger, Alexander Grill, Thomas Michalek

KNX Virtual 2.0 - User Interface Design



Projektauftraggeber*in: KNX Association
Projektbetreuer*in: Dipl.-Ing.in Patrizia Sailer, BSc MSc MBA
Projektmitglieder: Camila Frei, Patrick Kristen, Tobias Thurner, Christoph Ulreich
Repository: https://github.com/CUlreich-FHB/hochschule-burgenland-ppr-knx-virtual-2.0

*Ziel des Praxisprojekts KNX Virtual 2.0 - User Interface Design war es, einen Prototypen für eine virtuelle KNX-Lernumgebung zu schaffen, der modern und benutzerfreundlich ist. Die Anwendung ist für Elektriker*innen, Lernende sowie technisch interessierte Hobby-Anwender*innen gedacht und soll den Einstieg in die Gebäudeautomation vereinfachen. Mit einer benutzerfreundlichen Oberfläche lassen sich KNX-Komponenten in einem zweidimensionalen Grundriss durch Drag and Drop platzieren, einstellen und in unterschiedlichen Lernszenarien simulieren. Das Hauptaugenmerk wurde auf die anschauliche Darstellung der Zusammenhänge eines KNX-Systems gelegt. Auf diese Weise werden die elementaren Funktionen der Gebäudeautomation anschaulich und interaktiv dargestellt.*

Auftraggeber

Die KNX Association ist eine internationale Organisation mit Sitz in Brüssel, die den offenen Standard KNX für die Gebäudeautomation weltweit betreut und kontinuierlich weiterentwickelt. Mit dem KNX-Standard ist es möglich, Systeme wie Beleuchtung, Beschattung, Heizung, Lüftung, Klimatisierung und Sicherheitstechnik herstellerunabhängig zu vernetzen und zu steuern. Er findet weltweit Anwendung in Wohn- und Zweckgebäuden. Michael Critchfield übernahm die Rolle des Projektsponsors, während Jürgen Pannosch als Projektleiter und Hauptansprechpartner auf Seiten des Auftraggebers fungierte. Das gemeinsame Ziel war es, eine intuitive Lern- und Simulationsumgebung zu schaffen, die den Einstieg in die KNX-Gebäudeautomation erleichtert.

Projektbeschreibung

Ziel des Projekts war die Entwicklung eines interaktiven Prototyps mit Schwerpunkt auf Benutzerfreundlichkeit und der verständlichen Vermittlung von KNX-Grundlagen. Die Anwendung ermöglicht die Darstellung eines Gebäudes in einer zweidi-

mensionalen Vogelperspektive, in der verschiedene KNX-Komponenten wie Sensoren, Aktoren und Endgeräte per Drag-and-Drop platziert, miteinander verbunden und über ein Detailpanel konfiguriert werden können. Ergänzend wurden unterschiedliche Lernszenarien entwickelt, um typische Anwendungsfälle der Gebäudeautomation realitätsnah zu simulieren.

Zu Beginn des Projekts wurden mehrere Anforderungen definiert. Die Anwendung sollte eine intuitive und leicht verständliche Benutzeroberfläche bieten, die Nutzer und Nutzerinnen einfachen Zugang zur KNX-Welt ermöglicht. Gleichzeitig sollte die Applikation für die Zukunft einfach erweiterbar sein. Der Fokus lag dabei auf der Entwicklung eines UI/UX-Prototyps zur Visualisierung und Simulation von KNX-Funktionen, während eine Anbindung an reale KNX-Hardware oder die ETS-Software bewusst nicht Bestandteil des Projekts war.

Projektverlauf

Um das Projekt KNX Virtual 2.0 erfolgreich umzusetzen, haben wir dieses in verschiedene Arbeitspakete unterteilt, welche über fünf Sprints umgesetzt wurden. Nachdem bisher kein Mitglied unseres Teams ein grundlegendes Verständnis von KNX hatte, war dies auch das erste Ziel das wir uns gesetzt hatten. Zusätzlich dazu wollten wir dem Auftraggeber eine Grundidee liefern, wie die App aufgebaut sein könnte. Aus diesem Grund wurden auch im Laufe des ersten Sprints eine Vielzahl an Mock-Ups und Wireframes mit konkreten Designvorschlägen entwickelt.

Nach der Übergabe-Besprechung der ursprünglichen Mock-Ups mussten wir hier allerdings leider feststellen, dass das Konzept von KNX doch nicht so einfach zu erlernen ist, wie wir uns ursprünglich gedacht hatten. Begriffe wie „Aktoren“ oder „Sensoren“, vor allem aber das Zusammenspiel aller Komponenten, konnten von unserer Seite her nicht richtig eingeordnet werden und dementsprechend fehlte in den Mock-Ups diesbezüglich auch die fachliche Tiefe. Sie wurden daher auch nicht abgenommen und in den nächsten Sprint gezogen.

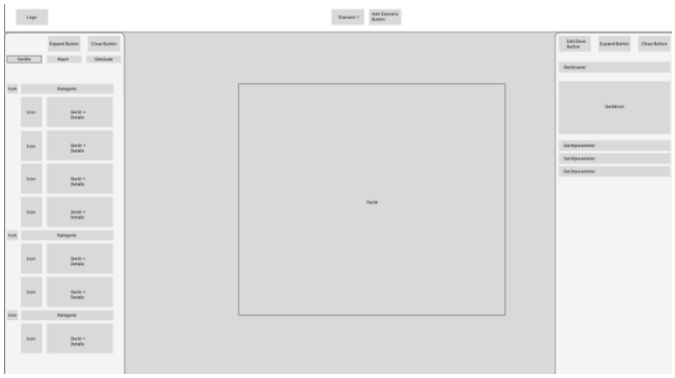


Abbildung 10: *Wireframe eines einfachen Raums*

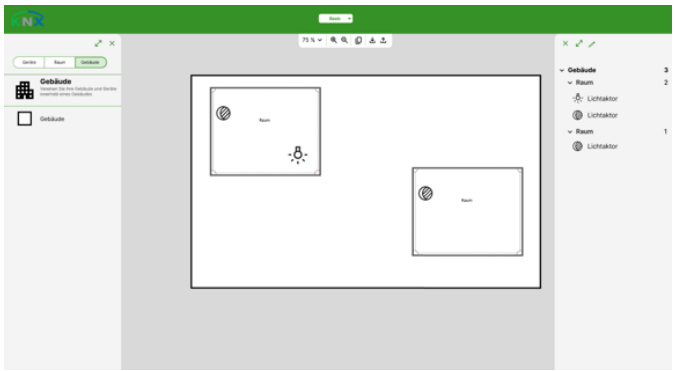


Abbildung 11: *Mock-Up eines Gebäudes mit zwei Räumen*

Sprint zwei widmete sich also ganz klar der Verbesserung der Mock-Ups sowie dem initialen Design der App. Um dies zu bewerkstelligen half uns Herr Pannosch nochmals mit einem KNX Crash-Kurs und erklärte uns Grundlagen, sowie die Funktionen und das Zusammenspiel der einzelnen Komponenten. Diese neuen Erkenntnisse wurden dann in die Mock-Ups eingearbeitet. Um jedoch nicht mit der konkreten Entwicklung hinterher zu hinken, wurde bereits das Grunddesign der App (Panels, Header, Workspace, etc.), sowie die ersten Geräte, parallel umgesetzt. Um dies zu bewerkstelligen, haben wir etwas eingeführt, das uns das gesamte restliche

Projekt begleitete, nämlich sehr klein geschnittene User-Stories. Diese halfen uns, in diesem und sämtlichen weiteren Sprints die App Schritt für Schritt aufzubauen und Themen extrem granular abzugrenzen.

In den nächsten Sprints widmeten wir uns dann der Erstellung und Erweiterung der am Beginn definierten Komponenten von Aktoren, Sensoren, Endgeräten, Wetterprofilen und Gebäudekomponenten, sowie der Entwicklung vorgefertigter Szenarien, als auch der konkreten KNX-Simulation.

Die Simulationslogik der Geräte war ein zentraler Bestandteil bzw. das Haupterkennungsmerkmal der App. Natürlich sollte es nicht nur möglich sein, einzelne Geräte in Räumen, Schaltschränken oder Gebäuden zu platzieren, sondern das Zusammenspiel und die Funktionalität dieser virtuell zu simulieren.

Aus diesem Grund führten wir eine Möglichkeit ein, die Geräte mittels KNX-Bus bzw. Stromkabel, je nach Gerätetyp, miteinander zu verbinden. Die konkrete Simulationslogik findet in den Sensoren statt. Dort ist es möglich festzulegen, welche Funktion der verbundenen Aktoren auf die verbundenen Endgeräte ausgeführt werden soll.

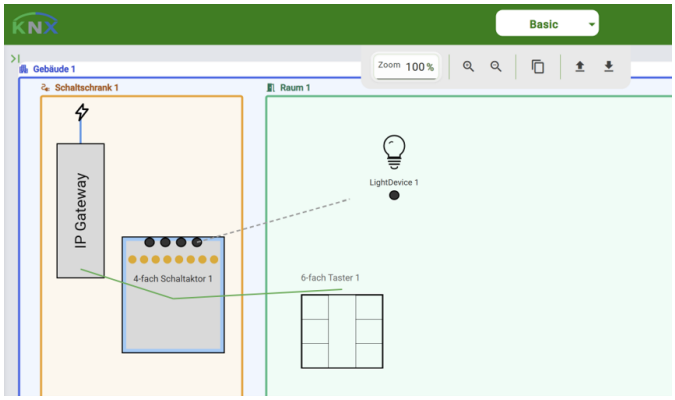


Abbildung 12: *Selbsterstelltes Lampenszenario*

Mit Anfang des letzten Sprints haben wir uns dann auf die bereits erwähnten vorgefertigten Szenarien konzentriert. Somit muss man sich die Geräte nicht mehr einzeln zusammenstöpseln, sondern kann mittels eines Buttonklicks konfigurierte und verbundene Geräte laden.

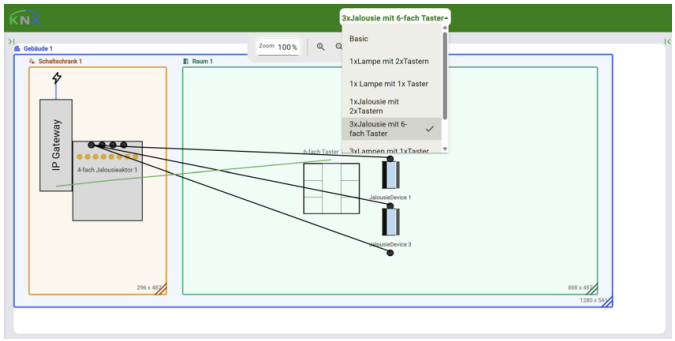


Abbildung 13: *Vordefiniertes Jalousieszenario*

Fazit

Die größte Schwierigkeit im Projekt war nicht die Technik, sondern KNX selbst zu verstehen. Unsere ersten Mock-Ups wurden nicht abgenommen, weil uns das Zusammenspiel von Aktoren, Sensoren und Endgeräten noch nicht klar genug war. Erst nach dem KNX-Crash-Kurs von Herrn Pannosch hatten unsere Entwürfe die nötige fachliche Tiefe.

Gut funktioniert hat dagegen, die User Stories sehr klein zu schneiden. Dadurch blieb der Fortschritt sichtbar und wir konnten Design und Entwicklung parallel vorantreiben. Auch die Entscheidung, sich auf den Prototyp zu beschränken und keine Hardware oder ETS anzubinden, hat sich als richtig erwiesen. So blieb genug Zeit für den Teil, auf den es ankam: die Simulation der KNX-Zusammenhänge.

Am Ende steht ein lauffähiger Prototyp, der die definierten Anforderungen erfüllt. Durch die klare Trennung von Geräten, Verbindungen und Szenarien lässt er sich gut erweitern. Sinnvolle nächste Schritte wären weitere Gerätetypen und Szenarien sowie ein Test mit echten Nutzer*innen aus der Zielgruppe.

Autor*innen: Camila Frei, Patrick Kristen, Tobias Thurner, Christoph Ulreich

Gamified Evidence-Based Nursing (GEBN)



Projektauftraggeber*in:	Gernot Heschl, Gesundheits- und Krankenpflege, Hochschule Burgenland
Projektbetreuer*in:	Dipl.-Ing. Franz Knipp
Projektmitglieder:	Rajeh Abdulhadi, Daniil Papertnyy, Felix Eibl
Repository:	https://gitlab.com/bswe-public/gebn

Im Praxisprojekt GEBN wurde ein analoger Escape Room aus der Lehrveranstaltung „Evidence-Based Nursing & Angewandte Pflegeforschung“ als Webanwendung digitalisiert, damit das Lehrformat orts- und zeitunabhängig einsetzbar ist. Studierende der GUK durchlaufen in Gruppen vier Spielräume, lösen Aufgaben zur evidenzbasierten Pflege (EBN) und sammeln Buchstaben für das Lösungswort FLORENCE. Umgesetzt wurde eine Full-Stack-Anwendung (React-Frontend, Spring-Boot-Backend) mit Admin-Dashboard für Lehrende, agil in fünf Sprints entwickelt. Drei der vier Räume wurden vollständig, der vierte als Mock, realisiert; der Funktionsumfang wurde zu rund 83 % erfüllt.

Auftraggeber

Auftraggeber ist Gernot Heschl aus dem Bereich GUK der Hochschule Burgenland. Als Fachexperte und Lehrender bringt er das didaktische Konzept ein, das dem Projekt zugrunde liegt: einen analogen Escape Room, mit dem er Studierenden das anspruchsvolle Thema der evidenzbasierten Pflege erfahrbar macht. In diesem Format bearbeiten die Studierenden Aufgaben zu Studienqualität, Evidenzstufen (LoE) und wissenschaftlicher Bewertung. Seine Rolle im Projekt war es, die fachlichen Anforderungen zu formulieren, die Inhalte der Aufgaben freizugeben und die Umsetzung in den Sprint Reviews aus Sicht der Lehre zu bewerten. Der enge fachliche Draht zum Auftraggeber war wichtig, weil die Korrektheit der pflege- und forschungsbezogenen Inhalte im Zentrum des Lernformats steht.

Projektbeschreibung

Gegenstand des Projekts war eine Webanwendung, die den analogen Escape Room vollständig digital abbildet. Bereits zu Projektbeginn standen die zentralen Vorgaben fest: vier aufeinander aufbauende Spielräume, die jeweils eine andere Kompetenz der evidenzbasierten Pflege abfragen. Der

Room of Knowledge ist ein Quiz mit Einzelfragen-Validierung, bei dem jede Frage korrekt beantwortet werden muss. Im Room of Abstracts ordnen die Gruppen wissenschaftlichen Abstracts die passende Evidenzstufe (LoE) zu. Der Room of Analytics verlangt analytisches Denken auf wissenschaftliche Daten, mit Echtzeit-Freigabe durch die Lehrenden. Im Room of Sciencebattle werden Studien verglichen und bewertet.

Für jeden abgeschlossenen Raum erhalten die Gruppen Buchstaben, aus denen sich am Ende das Lösungswort FLORENCE ergibt. Weitere feste Anforderungen waren ein Admin-Dashboard zur Verwaltung der Spielsitzungen, ein Beitritt per Code ohne Registrierung, ein Echtzeit-Countdown sowie ein responsives Design. Bewusst ausgeklammert wurden Registrierung, Chat, ein Frageneditor im UI und Internationalisierung; die Anwendung ist rein deutschsprachig. Technisch entstand ein Frontend mit React 19, Vite, TypeScript und Tailwind sowie ein Backend mit Spring Boot und Java 21 über eine REST-API. Lokal kam H2, für die Produktion PostgreSQL zum Einsatz; der Quellcode wird über GitLab mit CI/CD-Pipeline verwaltet.

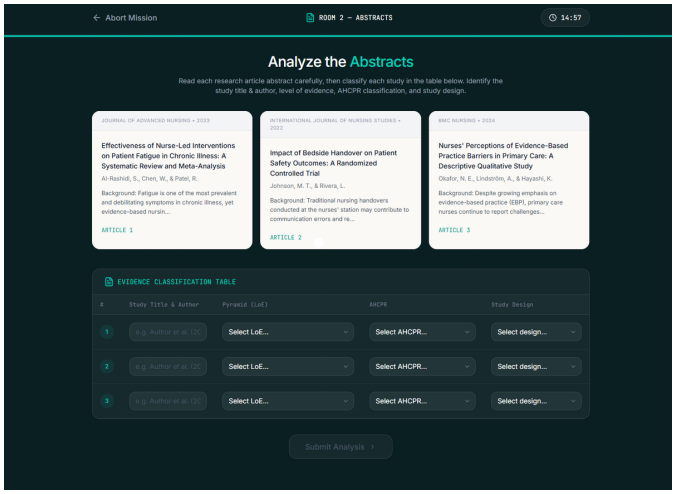


Abbildung 14: *Room of Abstracts: Zuordnung der Evidenzstufen (LoE) zu wissenschaftlichen Abstracts.*

Projektverlauf

Das Projekt gliederte sich in eine Planungs-, eine Umsetzungs- und eine Übergabephase und wurde in fünf zweiwöchigen Sprints umgesetzt. Nach dem Kick-Off am 06.03.2026 legte das Team die technische Grundlage: GitLab-Repositories für Frontend und Backend, ER-Modell und Datenbank, API-Spezifikation, ein UML-Use-Case-Diagramm, einen Prototyp sowie die statische Code-Analyse mit Pre-Commit-Hooks. Ein wichtiger Meilenstein war das durchgängige Erreichen von 80 % Testabdeckung in beiden Codebasen, das über das ganze Projekt als Qualitätsanker diente.

In der Umsetzungsphase entstanden Sprint für Sprint die vier Spielräume, der Raumübergang, das Beitritts- und Sitzungssystem sowie das Admin-Dashboard. Ein wiederkehrendes Muster war der Zweischritt aus Frontend-Mock und anschließender Ersetzung durch die reale Logik, wodurch frühe Demos möglich waren, ohne die Integration zu blockieren. Insgesamt lieferte das Team rund 195 Story Points aus.

Der Verlauf hatte auch seine Tiefen. Die größte Herausforderung war der Ausfall eines Teammitglieds: Raja Abdulhadi schied nach Sprint 1 aus, sodass die verbleibenden drei Mitglieder die offenen Aufgaben übernahmen und die Sprint-Ziele an die reduzierte Teamgröße anpassten. In der Folge wurde der Room of Sciencebattle bewusst zurückgestellt und nur als funktionsfähiger Mock umgesetzt; einige Komfortfunktionen des Dashboards wanderten in den Backlog. Technische Risiken, etwa bei der WebSocket-Integration des Countdowns und der Front-/Backend-Abstimmung, blieben dank früher API-Absprachen und konsequenter Code Reviews gering. Die Zeiterfassung zeigt, dass alle drei durchgängig aktiven Mitglieder das Semesterziel von rund 300 Stunden erreicht bzw. leicht übertroffen haben.

Den Abschluss bildete die Übergabephase. Das Team erstellte die technische Dokumentation so-

wie ein Benutzer*innenhandbuch für Lehrende und Studierende, übergab den lauffähigen Quellcode und bereitete die Abschlusspräsentation vor, die am 26.06.2026 an der Hochschule Burgenland stattfand. Parallel wurde das Testkonzept nach ISO 29119-3 finalisiert und die Testfälle wurden in XRAY dokumentiert, sodass die Qualitätssicherung bis zum Projektende lückenlos nachvollziehbar blieb.

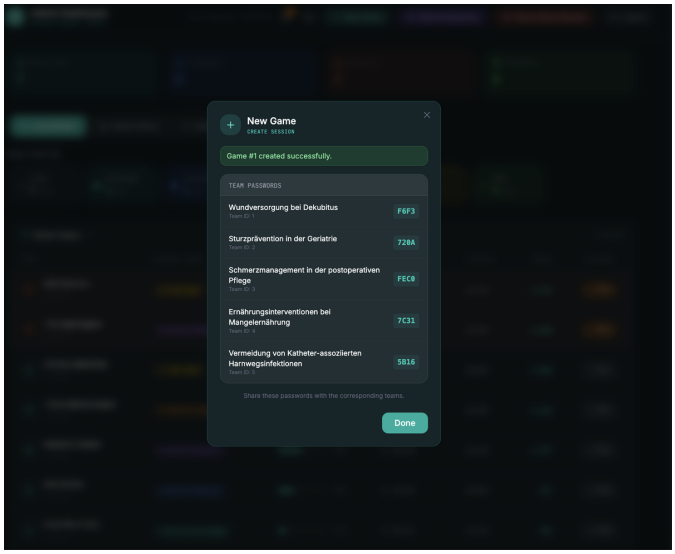


Abbildung 15: *Admin-Dashboard: Erstellung einer Spielsitzung mit Team-Passwörtern.*

Fazit

Das Praxisprojekt GEBN hat gezeigt, dass sich ein didaktisches Analog-Format mit vertretbarem Aufwand in eine funktionsfähige Webanwendung überführen lässt, wenn Anforderungen früh geklärt und Aufgaben klar geschnitten werden. Drei der vier Spielräume wurden vollständig, der vierte als Mock umgesetzt; mit einem Erfüllungsgrad von rund 83 % wurde der Kern des Spielkonzepts erfolgreich realisiert. Als wichtigste Erkenntnisse nimmt das Team mit: Die klare Aufteilung in fachliche Räume ermöglichte paralleles Arbeiten und eindeutige Verantwortlichkeiten. Durchgängige Nachvollziehbarkeit über Jira, XRAY und GitLab sowie regelmäßige Code Reviews hielten die Qualität hoch. Rückblickend hätte der Ausfall eines Mitglieds früher kompensiert und der Funktionsumfang einzelner Räume zu Projektbeginn enger geschnitten werden können. Nicht zuletzt hat sich die früh etablierte Testkultur ausgezahlt: Die durchgängige 80-Prozent-Testabdeckung gab dem Team die Sicherheit, auch unter Zeitdruck und mit reduzierter Teamgröße Änderungen vorzunehmen, ohne bestehende Funktionen zu gefährden. Der modulare Aufbau der vier Räume erlaubt es zudem, weitere Themen oder Fragensätze mit geringem Aufwand zu ergänzen: Die fachlichen Inhalte werden über einen Datenbank-Seed gepflegt und lassen sich so aktualisieren, ohne in den Code einzugreifen. Damit ist die Anwendung nicht auf den einmaligen Einsatz beschränkt, sondern kann in kommenden Semestern weiterverwendet und um zusätzliche Räume erweitert werden. Für die Leser*innen bleibt festzuhalten: Gamification ist kein Selbstzweck. Der Mehrwert von GEBN liegt darin, ein anspruchsvolles fachliches Thema so aufzubereiten, dass es Studierende motiviert und dass Lehrende es flexibel und ortsunabhängig einsetzen können.

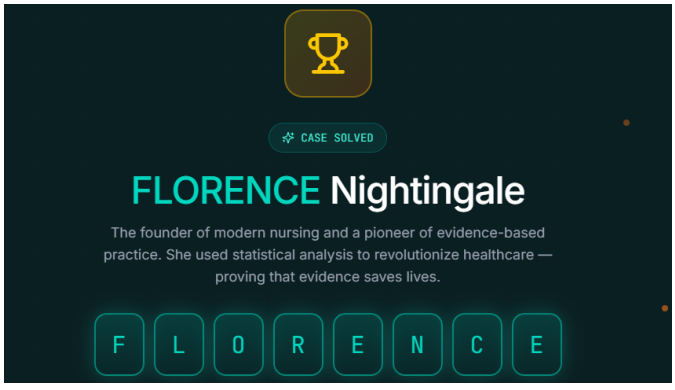


Abbildung 16: *Spielabschluss: das gelöste Lösungswort FLORENCE (Florence Nightingale).*

Autor*innen: Daniil Papertnyy, Felix Eibl, Rajeh Abdulhadi

Pegelhub – Connectoren und Funktionsmodule v6



Projektauftraggeber*in:	viadonau – Österreichische Wasserstraßen-Gesellschaft mbH
Projektbetreuer*in:	Dipl.-Ing.in Patrizia Sailer, BSc MSc MBA
Projektmitglieder:	Marcel Dumitru, Michael Mayr, Andreas Weinhofer
Repository:	https://github.com/orgs/Pegelhub-Praxisprojekt-SS26/repositories

Pegelhub ist eine Plattform zur Sammlung und Bereitstellung von Pegelmessdaten, die über mehrere Praxisprojekte der Hochschule Burgenland hinweg entwickelt wird. Im Sommersemester 2026 erweiterte Gruppe E das bestehende System um drei Komponenten: einen MQTT-Connector, der Wasserstandsdaten aus der Echtzeitdateninfrastruktur (EDIS) von PEGELONLINE empfängt und in den Pegelhub-Core einspeist, ein Qualitätssicherungsmodul, das die gesammelten Messwerte anhand konfigurierbarer Prüfregeln laufend überwacht, sowie einen Benachrichtigungsdienst, der bei erkannten Auffälligkeiten E-Mails versendet und SNMP-Traps auslöst. Alle Komponenten lassen sich über eigene Weboberflächen im laufenden Betrieb konfigurieren.

Auftraggeber

Auftraggeber des Projekts war die Donauhochwasserschutz-Konkurrenz (DHK), vertreten durch viadonau – Österreichische Wasserstraßen-Gesellschaft mbH als geschäftsführende Stelle. Die DHK besteht seit 1927 als Rechtsnachfolgerin der Donau-Regulierungs-Commission. Zu ihren Aufgaben zählen die Erhaltung und der Betrieb der Hochwasserschutz- und Dammbauten entlang der Donau sowie des Wiener Donaukanals. Ansprechpartner auf Auftraggeberseite war Johannes Groisz.

Verlässliche und aktuelle Pegelstände sind eine zentrale Grundlage für den Hochwasserschutz. Mit Pegelhub entsteht deshalb eine Plattform, die Pegelmessdaten aus unterschiedlichen Quellen zusammenführt und zentral bereitstellt. Der Pegelhub-Core wurde bereits in früheren Praxisprojekten entwickelt. Connectoren speisen Messdaten verschiedener Herkunft über eine einheitliche Schnittstelle ein. Aufgabe dieses Projekts war es, eine neue Echtzeit-Datenquelle anzubinden und das System um eine automatische Qualitätsüberwachung der eingehenden Daten mit aktiver Alarmierung zu ergänzen.

Projektbeschreibung

Ziel des Projekts war es, Pegelhub um Connectoren und Funktionsmodule zu erweitern. Der bestehende Pegelhub-Core war dabei als produktives System unverändert zu übernehmen; seine REST-Schnittstelle bildete die verbindliche Vorgabe, gegen die die neuen Komponenten entwickelt wurden.

Der MQTT-Connector bindet die Echtzeitdateninfrastruktur (EDIS) vom Pegelportal der deutschen Wasserstraßen- und Schifffahrtsverwaltung (WSV) an. Er abonniert die MQTT-Topics konfigurierbarer Pegelstationen, verarbeitet die eingehenden Nachrichten und leitet die enthaltenen Wasserstands-Messwerte an den Pegelhub-Core weiter. Zusätzlich veröffentlicht der Connector jeden verarbeiteten Messwert wieder, sodass nachgelagerte Anwendungen die Daten direkt beziehen können.

Das Qualitätssicherungsmodul überwacht die im Pegelhub-Core gesammelten Messwerte. Es ruft in konfigurierbaren Intervallen die Daten ausgewählter Datenquellen ab und prüft sie anhand von vier Prüfregeln: ob Werte einen zulässigen Bereich verlassen, ob aufeinanderfolgende Werte unplausibel springen, ob eine Messreihe über längere Zeit unverändert bleibt und ob Werte zu stark von einem Referenzwert abweichen. Jede Regel ist einzeln aktivierbar und parametrierbar. Aus den geprüften Werten kann das Modul außerdem einen virtuellen Pegel als Mittelwert mehrerer Stationen bilden und als eigenen Messpunkt in den Pegelhub-Core zurückschreiben.

Die Prüfergebnisse werden in die Kategorien gültig, Warnung und fehlgeschlagen eingeteilt. Für fehlgeschlagene Prüfungen erzeugt das Modul eine Alarmmeldung und übergibt sie an den Benachrichtigungsdienst. E-Mail- und SNMP-Alarmierung lassen sich dabei unabhängig voneinander aktivieren und deaktivieren.

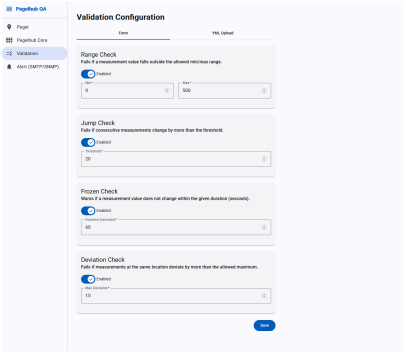


Abbildung 17: *Konfiguration der Prüfregeln im Qualitätssicherungsmodul*

Der Benachrichtigungsdienst versendet bei erkannten Qualitätsverletzungen Alarmmeldungen als E-Mail an einen konfigurierbaren Empfängerkreis sowie als SNMP-Trap an Monitoring-Systeme (wahlweise über SNMPv2c oder das benutzerbasierte Sicherheitsmodell von SNMPv3). Betreff, Text und Signatur der Nachrichten sowie Trap-Ziele und Objekt-Identifikatoren (OIDs) sind frei konfigurierbar. Alle drei Komponenten verfügen darüber hinaus über eine eigene Management-Oberfläche, über die sich sämtliche Einstellungen im Browser ändern lassen (zur Laufzeit, wahlweise per Formular oder YAML-Upload).

Projektverlauf

Zu Projektbeginn stand die Einarbeitung in das bestehende Pegelhub-System und in die EDIS-Datenwelt. Es war notwendig, zu den Technologien SNMP und SMTP Wissensaneignung zu betreiben und sowohl die Dictionary-API von PEGELONLINE und die Pegelhub-API als auch die uns zur Verfügung gestellten Testserver zu analysieren.

Gearbeitet wurde iterativ in Sprints. Anforderungen wurden als User-Stories in Jira verwaltet und über XRay mit Testfällen verknüpft, sodass die Nachvollziehbarkeit von der Anforderung bis zum Testergebnis durchgehend gegeben war. In allen Repositories galten einheitliche Quality-Gates: Formatierungs- und statische Codeprüfungen lie-

fen über Git-Hooks vor jedem Commit sowie in der GitLab-CI-Pipeline bei jedem Push.

Bei der Umsetzung des MQTT-Connectors galt es neben dem eigentlichen Datenfluss die Robustheit der Verarbeitung zu garantieren: Fehlerhafte oder unvollständige Nachrichten werden aussortiert, ohne den Empfang zu unterbrechen, und nach Konfigurationsänderungen über die Weboberfläche baut der Connector seine Broker-Verbindungen automatisch neu auf.

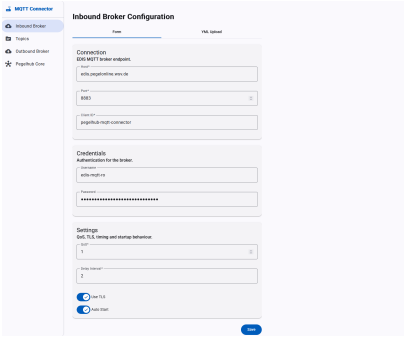


Abbildung 18: *Management-Oberfläche des MQTT-Connectors*

Beim Qualitätssicherungsmodul entfiel der größte Teil der Arbeit auf den Entwurf der Prüfregeln. Da sinnvolle Schwellwerte, etwa für Sprünge zwischen aufeinanderfolgenden Werten oder die Dauer, ab der eine unveränderte Messreihe als auffällig gilt, je Pegel unterschiedlich sind, wurden alle Regeln parametrierbar ausgelegt. Ein länger unentdecktes Missverständnis betraf das Datenmodell des Pegelhub-Core: Kennungen, die zunächst als Messgrößen interpretiert wurden, bezeichnen tatsächlich registrierte Datenquellen. Zur Korrektur war es auch notwendig Konfigurationsschlüssel und die Dokumentation anzupassen. Das zeigt, wie wichtig es ist, die Begriffe einer fremden Schnittstelle früh gegen das eigene System abzugleichen.

Für den Benachrichtigungsdienst war eine Einarbeitung in SNMP erforderlich, mit dem zuvor niemand im Team gearbeitet hatte, insbesondere in Trap-Formate, OID-Strategien und die Sicherheitsstufen von SNMPv3, von einfachen Community-Strings bis zur authentifizierten und verschlüsselten Übertragung. Der Dienst lässt sich dadurch an unterschiedliche Monitoring-Umgebungen anpassen.

Die letzte Projektphase galt der Integration der getrennt entwickelten Komponenten zu einem konsistenten Gesamtsystem: Ports und Konfigurationsschnittstellen wurden vereinheitlicht, die drei Angular-Oberflächen auf ein identisches Layout gebracht und jedes Frontend direkt in das zugehörige Spring-Boot-Backend gebündelt. Parallel dazu entstanden eine technische Dokumentation und ein Benutzerhandbuch.

Fazit

Die wichtigste Erkenntnis des Projekts: Die eigentliche Schwierigkeit lag nicht im Entwickeln der einzelnen Komponenten, sondern im Zusammenspiel mit dem bestehenden Pegelhub-Core, dessen Schnittstelle wir als fixen Vertrag übernehmen mussten, und zwischen den eigenen, zunächst getrennt entwickelten Diensten.

Bewährt haben sich die von Anfang an etablierten Quality-Gates und die durchgängige Nachvollziehbarkeit von der User-Story bis zum Testergebnis. Ebenso bewährt hat sich die Entscheidung, die Konfiguration aller Dienste zur Laufzeit über die Weboberflächen änderbar zu machen. Das System lässt sich im Betrieb anpassen, ohne dass dafür Entwicklerwissen nötig ist.

Am Ende steht ein lauffähiges Gesamtsystem: Pegelhub empfängt Echtzeitdaten aus EDIS, überwacht deren Qualität automatisch und alarmiert aktiv bei Auffälligkeiten. Mit der Dokumentation ist die Grundlage gelegt, auf der ein Folgeprojekt direkt weiterarbeiten kann.

Autoren: Marcel Dumitru, Michael Mayr, Andreas Weinhofer

Projektmitglieder – BSWE Jahrgang 2024

Weiterentwicklung gAmIcode



Aurika Afanaseva
Hochschule Burgenland



Nicolas Haselbacher
Hochschule Burgenland



Jeremy Nicolas Heißenberger
Hochschule Burgenland



Marco Magyar
Hochschule Burgenland

CourseGuide



Alexander Brenner
Hochschule Burgenland



Süleyman Flachberger
Hochschule Burgenland

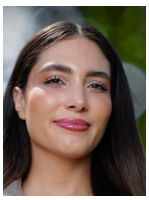


Alexander Grill
Hochschule Burgenland



Thomas Michalek
Hochschule Burgenland

KNX Virtual 2.0 - User Interface Design



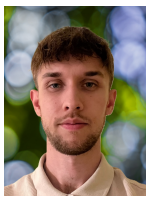
Camila Frei
Hochschule Burgenland



Patrick Kristen
Hochschule Burgenland



Tobias Thurner
Hochschule Burgenland

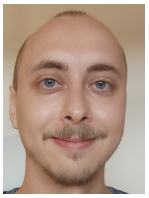


Christoph Ulreich
Hochschule Burgenland

Gamified Evidence-Based Nursing (GEBN)



Daniil Papertnyy
Hochschule Burgenland



Felix Eibl
Hochschule Burgenland



Rajeh Abdulhadi
Hochschule Burgenland

Pegelhub – Connectoren und Funktionsmodule v6



Andreas Weinhofer
Hochschule Burgenland



Michael Mayr
Hochschule Burgenland



Marcel Dumitru
Hochschule Burgenland

Projektbetreuer*innen



Dipl.-Ing. Franz Knipp

Studiengangsleiter des Bachelorstudiengangs Software Engineering und vernetzte Systeme an der Hochschule Burgenland



Dipl.-Ing.in Patrizia Sailer, BSc MSc MBA

Fachhochschullehrerin am Department für Informationstechnologie an der Hochschule Burgenland



Julia Pausch, BSc MSc MA

Externe Lektorin für die Lehrveranstaltung Projektmanagement



Renate Weichselbraun

Externe Lektorin für die Lehrveranstaltung Softwarequalität



Hans Sowa, BSc

Externer Lektor für die Lehrveranstaltung Softwarequalität

Verzeichnisse

Abbildungsverzeichnis

1 Komponenten und Kommunikationswege der finalen gAmIcode-Systemarchitektur . . . 3
2 Maven-Testausgabe des Java-Runners im Code-Editor . . . 4
3 Live-Vorschau einer HTML-Aufgabe mit Nodepod . . . 4
4 SQL-Abfrage mit Ergebnistabelle des SQL-Runners . . . 4
5 Lektorenansicht mit wirksamen Maven- und Parsing-Einstellungen . . . 5
6 Das Ampelsystem in Grün, Gelb und Rot . . . 7
7 Warnstatus bei nicht sichtbarer Literaturliste . . . 7
8 Detailansicht mit Validierungsfehlern . . . 8
9 Vollständig validierter Kurs . . . 8
10 Wireframe eines einfachen Raums . . . 10
11 Mock-Up eines Gebäudes mit zwei Räumen . . . 10
12 Selbsterstelltes Lampenszenario . . . 11
13 Vordefiniertes Jalousieszenario . . . 11
14 Room of Abstracts: Zuordnung der Evidenzstufen (LoE) zu wissenschaftlichen Abstracts. 13
15 Admin-Dashboard: Erstellung einer Spielsitzung mit Team-Passwörtern. . . . 14
16 Spielabschluss: das gelöste Lösungswort FLORENCE (Florence Nightingale). . . . 14
17 Konfiguration der Prüfregeln im Qualitätssicherungsmodul . . . 16
18 Management-Oberfläche des MQTT-Connectors . . . 17

Abkürzungsverzeichnis

API	Application Programming Interface.
API	Application Programming Interface.
CI	Continuous Integration.
CI/CD	Continuous Integration und Continuous Delivery.
E2E	End-to-End.
E2E	End-to-End.
EBN	Evidence-Based Nursing.
ECTS	European Credit Transfer and Accumulation System.
GUK	Gesundheits- und Krankenpflege.
IDE	Integrated Development Environment.
IT	Informationstechnologie.
LoE	Level of Evidence.
LTI	Learning Tools Interoperability.
PoC	Proof of Concept.
REST	Representational State Transfer.
UI	User Interface.
UX	User Experience.

Impressum

Hochschule für Angewandte Wissenschaften Burgenland GmbH
Studienzentrum Eisenstadt | Campus 1 | 7000 Eisenstadt
Studienzentrum Pinkafeld | Steinamangerstraße 21 | 7423 Pinkafeld

Tel.: +43 5 7705

E-Mail: office@hochschule-burgenland.at | **www.hochschule-burgenland.at**

Dieses Werk ist lizenziert unter einer Creative Commons Namensnennung - Nicht-kommerziell - Weitergabe unter gleichen Bedingungen 4.0 International Lizenz. Diese Lizenz erlaubt es, den Inhalt zu vervielfältigen, zu verbreiten und öffentlich aufzuführen unter folgenden Bedingungen: Der Name des Autors/Rechtsinhabers muss genannt werden. Dieser Inhalt darf nicht für kommerzielle Zwecke verwendet werden. Bearbeiteter oder in anderer Weise umgestalteter Inhalt darf als neu entstandener Inhalt nur unter Verwendung identischer Lizenzbedingungen weitergegeben werden. http://creativecommons.org/licenses/by-nc-sa/4.0/

This work is licensed under Attribution-NonCommercial-ShareAlike 4.0 International. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-sa/4.0/