weblablib: Ein neuer Ansatz zur Einrichtung von Remote-Laboren

Pablo Orduña, Luis Rodriguez-Gil, Ignacio Angulo, Unai Hernandez, Aitor Villar, Javier Garcia-Zubia

Abstract

Remote-Labore bestehen aus Hard- und Softwarekomponenten, die es Studierenden ermöglichen, über das Internet auf reale Geräte zuzugreifen. Remote-Labor-Management-Systeme (RLMS) sind Software-Tools, die speziell dafür entwickelt wurden, Remote-Labore auf einfachere Weise einzurichten. RLMS bieten dafür Querschnittsfunktionen, die inzwischen in den meisten Remote-Laboren üblich sind (z. B. Authentifizierung, Autorisierung, Zugriffsmanagement oder Administration), sowie einige für den Zugang zu den Laboren notwendige Protokolle oder APIs (Application Programming Interfaces). WebLab-Deusto ist ein verbreitetes Open-Source-RLMS, das in verschiedenen Hochschulen zur Einrichtung und Administration von Remote-Laboren verwendet wird und dabei zwei Ansätze verfolgt: *gemanagt* (die gesamte Kommunikation läuft über WebLab-Deusto) und *nicht gemanagt* (die Kommunikation wird von den Betreibenden des Remote-Labors selbst verwaltet). Der *gemanagte* Ansatz besaß ursprünglich eine Reihe von Vorteilen gegenüber dem *nicht gemanagten*. Doch im Laufe der Zeit wurde es notwendig, passenden Support auch für *nicht gemanagte* Remote-Labore anzubieten, da sich die Webentwicklungstechnologien heutzutage schnell ändern und ihre Leistungsfähigkeit kontinuierlich zunimmt. Vor diesem Hintergrund entwickelte LabsLand das Open-Source-Framework *weblablib*. Der vorliegende Artikel stellt dieses neue Framework ebenso wie die Herausforderungen vor, mit denen sich die Entwickler*innen von Remote-Laboren im Zuge der Implementierung konfrontiert sehen.

Schlüsselwörter: Remote-Labor, Remote-Labor-Management-Systeme (RLMS), weblablib, WebLab-Deusto

1 Einleitung

Ein Remote-Labor im Lehr-Lernkontext ist eine Software- und Hardwarelösung, die es Studierenden ermöglicht, über einen Standard-Webbrowser so auf reale Laboreinrichtungen ihrer Hochschule zuzugreifen, als befänden sie sich vor Ort. In der Regel werden solche Labore von den Hochschulen selbst oder von Forschungszentren betrieben.

Ein Schlüsselfaktor von Remote-Laboren besteht darin, dass sich ihre Nutzungsmöglichkeiten deutlich erweitern, sobald sie nicht nur im hochschuleigenen

Intranet, sondern auch über das Internet verfügbar sind und somit ebenfalls von Lernenden anderer Einrichtungen genutzt werden können. So können sich zwei oder mehr Institutionen Equipment teilen, das häufig nur wenige Stunden am Tag oder an bestimmten Tagen im Jahr genutzt wird, und ihre Kosten dadurch senken. Darüber hinaus wird dadurch der Einstieg in eine Sharing Economy befördert, in der mehrere Einrichtungen einander gegenseitig – kostenfrei oder gegen Entgelt – den Zugang zu ihren Laboren gewähren.

Die Forschungsliteratur beschreibt eine große Vielfalt an Remote-Laboren, etwa in den Bereichen Robotik, Elektronik, Physik oder Chemie. Spezielle Software-Frameworks (z. B. RLMS wie WebLab-Deusto[1] [Orduña et al., 2014], iLab Shared Architecture[2], RemLabNet[3] [Schauer et al., 2016] und Labshare Sahara[4] [Lowe, Machet & Kostulski, 2012]) machen deren Entwicklung erschwinglicher und Tools wie gateway4labs[5] (Orduña et al., 2015) ermöglichen ihre Integration in Learning-Management-Systeme (LMS) wie Moodle oder Sakai – sowohl mittels Ad-hoc-Lösungen als auch durch Standards wie IMS LTI[6]. Darüber hinaus wurden Repositorien geschaffen, die Remote- und virtuelle Labore miteinander verbinden (z. B. Go-Lab [de Jong, Linn & Zacharia, 2013; Gillet, de Jong, Sotirou & Salzmann, 2013], LiLa [Richter, Boehringer & Jeschke, 2011] und iLabCentral).

Den meisten der genannten Technologien liegt die Idee zugrunde, Remote-Labore im Hinblick auf ein gemeinsames Wachstum zu nutzen: Bei gemeinsamer Nutzung stünden zwei Hochschulen mit jeweils drei eigenen Remote-Laboren theoretisch insgesamt sechs Labore zur Verfügung. Dieses Szenario ist allerdings mit einer Reihe organisatorischer Herausforderungen verbunden, z. B. im Hinblick auf die Betriebszuverlässigkeit oder das Nutzer*innenvertrauen. Wie können sich beispielsweise die Lehrenden sicher sein, dass die Remote-Labore der jeweils anderen Hochschule auch mehrere Jahre Bestand haben? In kleinen Verbünden, z. B. innerhalb eines geförderten Projekts oder in langjährigen, intensiven Partnerschaften zwischen Hochschulen, lassen sich in der Regel Antworten finden – in einem größeren Kontext werden diese Herausforderungen jedoch schnell zum Problem.

Vor diesem Hintergrund und mit dem erklärten Ziel, den Bereich Remote-Labore auf eine neue Ebene zu heben, gründete das Team von WebLab-Deusto *LabsLand*[7] als Spin-off der Universität Deusto (Bilbao, Spanien), um aktuellen und künftigen Netzwerkpartnern jederzeit die erforderliche Betriebszuverlässigkeit sowie ein professionelles wirtschaftliches Fundament zu garantieren. Mehrere Institutionen in verschiedenen Ländern stellen bereits Ressourcen für die Einrichtung neuer Remote-Labore zur Verfügung oder nutzen die gut geführten Labore des LabsLand-

1 http://weblab.deusto.es.
2 http://ilab.mit.edu.
3 http://www.remlabnet.eu.
4 https://remotelabs.eng.uts.edu.au.
5 http://gateway4labs.readthedocs.org.
6 Learning Tools Interoperability (LTI) ist ein vom IMS (Instructional Management Systems) Global Learning Consortium entwickelter Standard für Bildungstechnologie.
7 https://labsland.com.

Netzwerks. Dabei kommen verschiedene RLMS zum Einsatz, nicht nur WebLab-Deusto.

Für den Erfolg der Idee hinter LabsLand ist es jedoch notwendig, dass im Netzwerk eine Vielzahl von Remote-Laboren zur Verfügung steht, die ein breites Spektrum an Fachgebieten abdecken. Um dies sicherzustellen, wurden Tools zur schnelleren und zuverlässigeren Einrichtung von Remote-Laboren entwickelt. Während einige von ihnen proprietär sind, handelt es sich bei *weblablib* um ein neuartiges Open-Source-Framework zur Einrichtung von Remote-Laboren, das auf WebLab-Deusto basiert.

Der Schwerpunkt des vorliegenden Beitrags liegt auf der Beschreibung von weblablib. Zu diesem Zweck werden zunächst die Grundlagen von RLMS und im Besonderen von WebLab-Deusto thematisiert und es wird geschildert, welchen Einschränkungen das Design von WebLab-Deusto vor der Entwicklung von weblablib unterlag. Schließlich richtet sich der Fokus auf weblablib selbst, um dessen Funktionen, Einsatzmöglichkeiten und Grenzen zu beschreiben.

2 Aktuelle Lösungen zur gemeinsamen Nutzung von Remote-Laboren

In diesem Abschnitt werden die Konzepte von Remote-Laboren, RLMS, Remote-Labor-Verbünden und Portalen zur gemeinsamen Nutzung von Remote-Laboren vorgestellt.

2.1 Remote-Labore

Ein Remote-Labor ist eine Hardware- und Softwarelösung, die es Studierenden ermöglicht, über das Internet auf reale Geräte in ihrer Hochschule zuzugreifen, als wären sie vor Ort im Labor. Abbildung 1 veranschaulicht dieses Prinzip: Sie zeigt ein mobiles kostengünstiges Roboterlabor, wie es in Pumezo et al. (2020) und Buitrago et al. (2020) beschrieben wird.

Die Studierenden lernen, einen Arduino-Microcontroller zu programmieren, schreiben zu Hause den Code, kompilieren ihn mit passenden Werkzeugen und senden die binäre Datei über das Internet an den realen Roboter. Anschließend können sie über das Internet beobachten, wie sich der Roboter auf Basis ihres Programms in einer realen Umgebung verhält und er z. B. wie gewünscht der schwarzen Linie folgt. In der Forschungsliteratur finden sich zahlreiche Beispiele für ein solches Verfahren (Gomes & Bogosyan, 2009; Gravier et al., 2008).

Tatsächlich wurden erste Remote-Labore schon vor mehr als zwei Jahrzehnten entwickelt (Carisa et al., 1995; Aktan et al., 1996; Henry, 1996). Seither fanden sie Anwendung in einer Vielzahl von Kontexten: Chemie (Coble et al., 2010; Cedazo et al., 2006), Physik (Del Alamo et al., 2002; Gillet et al., 2001), Elektronik (Gustavsson et al., 2007; Nedic, Machotka & Nafalski, 2008), Robotik (Šafarič et al., 2005; Torres et al., 2006), Akustik (Zappatore, Longo & Bochicchio, 2016) und sogar Nukleartechnik (Hardison et al., 2008).

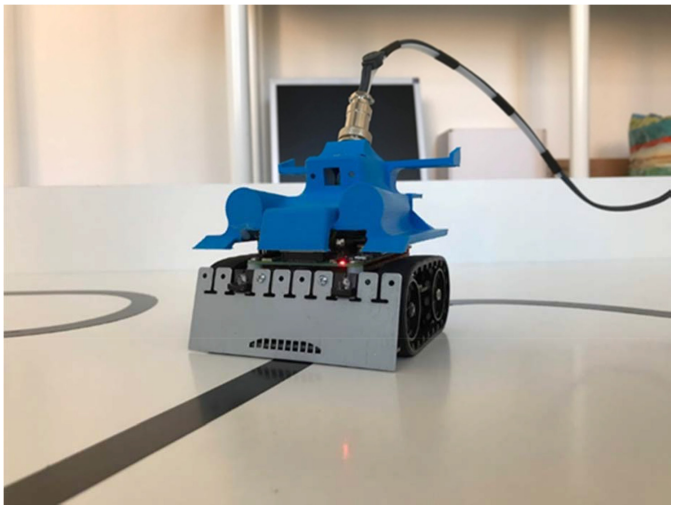

Abbildung 1: Roboterlabor zur Programmierung eines Arduino (Pumezo et al., 2020; Buitrago et al., 2020)

2.2 Remote-Labor-Management-Systeme

Jedes gängige Remote-Labor verwendet, unabhängig von seinem speziellen Einsatzgebiet, zumindest mehrere der folgenden allgemeinen Systemfunktionen: Authentifizierung, Autorisierung, Zugriffsmanagement (typischerweise mittels Warteschlange oder terminbasierten Buchungen), User Tracking und Administration. So finden sich beispielsweise ein Authentifizierungswerkzeug und ein Warteschlangensystem sowohl in Elektronik- als auch in Chemielaboren.

Remote-Labor-Management-Systeme (RLMS) setzen genau hier an – u. a. MIT iLabs[8], WebLab-Deusto[9] oder Labshare Sahara[10]. Sie stellen Toolkits zur Einrichtung neuer Remote-Labore ebenso wie Management-Tools und allgemeine Dienste für Remote-Labore zur Verfügung (z. B. zur Authentifizierung, Autorisierung oder Zeitplanung). Dabei besteht die Grundidee darin, dass alle Labore, die vom RLMS verwaltet werden, bei Hinzufügen einer neuen Funktion (etwa Zugriff per LDAP [Lightweight Directory Access Protocol] oder Learning Analytics Panel [Orduña et al., 2014]) automatisch die jeweilige Funktion unterstützen.

2.3 Zusammenschluss von Remote-Laboren

Wie eingangs erwähnt, besteht ein Schlüsselaspekt von Remote-Laboren darin, dass sie von verschiedenen Institutionen gemeinsam genutzt werden können, sobald sie über das Internet verfügbar gemacht wurden. Insgesamt gibt es drei Ansätze zur Nutzung von Remote-Laboren:

- Offene Nutzung: Der Zugang zu den Laboren bleibt völlig offen und die Labore sind frei nutzbar. Dies geht jedoch womöglich zulasten der zur Verfügung ge-

8 http://ilab.mit.edu.
9 http://weblab.deusto.es.
10 http://github.com/saharalabs

stellten Learning Analytics und der Nachverfolgbarkeit bzw. Zuordnung. Auch kann es in diesem Rahmen schwierig werden, die Nutzung des Remote-Labors durch Studierende verschiedener Institutionen zu priorisieren. Das Ergebnis ist ein Kompromiss zwischen freier Nutzbarkeit und erweiterter Funktionalität (Orduña et al., 2015).

- Gegenseitiges Bereitstellen von Laboren: Es erfolgt eine gemeinsame Nutzung von Konten zwischen den verschiedenen RLMS. Möchte Hochschule A Labore von Hochschule B nutzen, stellt Hochschule A Hochschule B eine Liste mit User-Namen zur Verfügung. Die Studierenden greifen daraufhin mit den Anmeldedaten von Hochschule B auf das System zu. Im Idealfall wird eine gemeinsame Authentifizierung verwendet, um die Bereitstellung von Anmeldedaten in verschiedenen Domänen (Shibboleth, OAuth o. Ä.) zu vermeiden. Dies ist normalerweise jedoch nicht der Fall.
- Laborverbund: Unterstützt ein RLMS einen Laborverbund und ist das System an zwei Hochschulen eingerichtet (z. B. an Hochschule A und Hochschule B), greifen die Studierenden der Hochschule A auf das RLMS der Hochschule A zu und nutzen durchlässig die Labore der Hochschule B. Sie arbeiten also in einem Institutionsverbund, sodass Hochschule B keine Liste der Studierenden von Hochschule A benötigt, sondern sich auf die gemeinsame hochschulübergreifende Vereinbarung verlassen kann.

Der fortschrittlichste unter den oben genannten Ansätzen ist der Zusammenschluss von Remote-Laboren durch Protokolle, die sich an marktähnlichen Situationen orientieren. Dabei werden Verbundprotokolle zur Förderung der Interoperabilität zwischen RLMS verwendet (Orduña et al., 2014). Diese kompatiblen Verbindungen zwischen verschiedenen Systemen können weiter verbessert werden, wenn Eigenschaften wie Transitivität oder gekoppelte Lastverteilung zur Verfügung stehen (Orduña, 2013).

3 WebLab-Deusto-Software

WebLab-Deusto ist ein Open-Source-RLMS, das die Entwicklung von Remote-Laboren ermöglicht – frei von den normalerweise damit verbundenen Problemen und Herausforderungen. Insbesondere bietet es folgende Funktionen:

- innovativer Warteschlangenmechanismus, der sowohl einen lokalen als auch einen gekoppelten Lastausgleich unterstützt (die Warteschlange wird z. B. automatisch aufgeteilt, wenn zwei Instanzen desselben Labors existieren und verfügbar sind)
- verschiedene Administrationstools und Dashboards für Learning Analytics (Administrator*innen und Lehrende können nachvollziehen, wer das Labor wie verwendet hat und auf welche Komponenten zugegriffen wurde)

- gemeinsame Nutzung von Laboren mit anderen WebLab-Deusto-Systemen anderer Institutionen – ohne Verwendung geteilter Zugangsdaten oder Duplizierung von Profilen
- verschiedene Authentifizierungsmechanismen; über gateway4labs erfolgt darüber hinaus Unterstützung für LTI, sodass WebLab-Deusto auch integriert in Learning Management Systems (LMS) genutzt werden kann
- native Unterstützung im LabsLand-Netzwerk sowie im Smart Gateway des Go-Lab-Projekts (Orduña et al., 2015)

WebLab-Deusto wurde auf Basis folgender Überlegungen konzipiert:

- Die Entwickler*innen von Remote-Laboren kommen aus den unterschiedlichsten Arbeitsgebieten und weisen ein ausgesprochen heterogenes Kompetenzprofil auf. Daraus resultieren auch durchaus divergierende Vorlieben für verschiedene Technologien. Im Hinblick auf eine breite Akzeptanz muss WebLab-Deusto deshalb mehrere Technologien unterstützen.
- Es gibt Entwickler*innen von Remote-Laboren, die keine umfassenden IT-Kenntnisse besitzen und es sollte ihnen auch nicht zugemutet werden, sich intensiver mit der Materie zu befassen. Für sie sollte WebLab-Deusto ein gewisses Abstraktionslevel bereitstellen. Andere Entwickler*innen wiederum besitzen ausgeprägte IT-Skills und sind in der Lage, uneingeschränkt mit den neuesten Technologien zu arbeiten.

Aus den oben geschilderten Gründen kamen bei der Entwicklung von WebLab-Deusto zwei Ansätze zum Tragen: Der *gemanagte* und der *nicht gemanagte* Ansatz.

Der *gemanagte Ansatz* ermöglicht die Entwicklung von Remote-Laboren, ohne dass sich die Verantwortlichen dabei im Detail mit Netzwerken oder Sicherheitssystemen auseinandersetzen müssen. Er baut auf einem Grundmodell auf, bei dem ein Code auf Client-Seite (z. B. in JavaScript) Befehle oder Meldungen über ein API sendet, ohne Kommunikationsprotokolle berücksichtigen zu müssen. WebLab-Deusto gewährleistet, dass diese Befehle oder Meldungen bis zum Experiment gesendet werden, wo ein anderer Code mittels einer einfachen Schnittstelle zur Programmierung von Anwendungen (API) ausgeführt wird. Die Entwickler*innen müssen nicht wissen, wie viele Instanzen des Remote-Labors es gibt, wo sie sich befinden, wie Zugriff und Zertifizierung konfiguriert sind usw. – sie senden und empfangen lediglich Daten. Das Protokoll wurde in vielen Programmiersprachen implementiert, darunter Python, C, C++, LabVIEW (Code), .NET, Java und Node.js. Remote-Labor-Entwickler*innen, die mit einem dieser Systeme vertraut sind, können somit direkt mit der Implementierung ihres Labors beginnen.

Der *nicht gemanagte Ansatz* bietet eine Web-Schnittstelle zur Einrichtung von Remote-Laboren. WebLab-Deusto gibt lediglich diese Schnittstelle vor, Bereitstellung, Adressierung, Datenschutz etc. liegen danach in der Verantwortung der Remote-Labor-Entwickler*innen, was für ein einfaches Remote-Labor im Vergleich zum gemanagten Ansatz ein erhebliches Mehr an Aufwand bedeutet. Der nicht ge-

managte Ansatz bietet versierten Entwickler*innen jedoch eine flexibel anpassbare Schnittstelle, die in jedem Web-Framework erweitert und mit jeder Webtechnologie verwendet werden kann.

Abbildung 2 zeigt die grundlegende lokale Architektur von WebLab-Deusto am Beispiel einer einzelnen Hochschule im Vergleich zur Verbundstruktur.

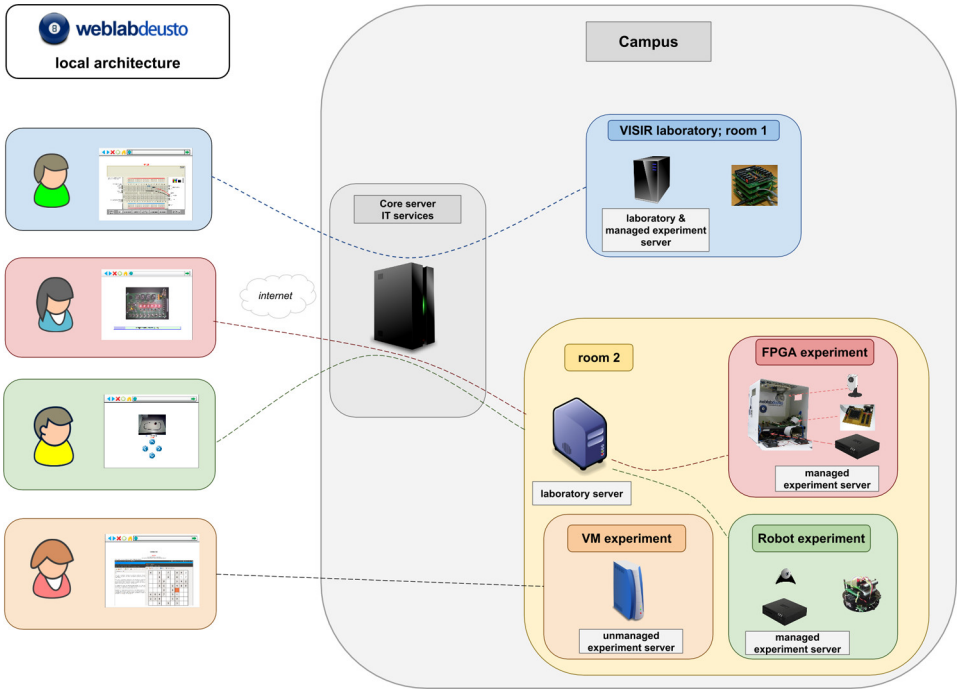

Abbildung 2: Aufbau von WebLab-Deusto

Verschiedene User greifen über dieselbe WebLab-Deusto-Schnittstelle auf verschiedene Labore an unterschiedlichen Standorten der Hochschule zu. Dies ermöglicht es den Entwickler*innen von Remote-Laboren, einen oder mehrere WebLab-Deusto-„Kernserver“ (betreut von den jeweiligen IT-Abteilungen) auf dem Campus zu betreiben und verschiedene Remote-Labore, die kostengünstige Technologien wie Raspberry Pi nutzen können, an unterschiedlichen Standorten einzusetzen. In den gemanagten Laboren werden sämtliche Befehle bzw. Meldungen über die „Kern-“ oder „Laborserver“ gesendet, während bei den nicht gemanagten Laboren lediglich der Reservierungsprozess über WebLab-Deusto erfolgt. Im Anschluss werden die User direkt an das entsprechende Labor weitergeleitet.

4 weblablib

Wie im vorangegangenen Abschnitt erläutert, unterstützt WebLab-Deusto sowohl den *gemanagten* als auch den *nicht gemanagten Ansatz*. Beim gemanagten Ansatz werden Bibliotheken für verschiedene Sprachen bereitgestellt. Beim nicht gemanagten Ansatz ist dies unmöglich, da die von Entwickler*innen hierfür potentiell genutzten Web-Frameworks zu vielfältig sind und sich zu häufig ändern. Das Web-Interface ist aus diesem Grund bewusst quelloffen gehalten, sodass Entwickler*innen anderer Frameworks es implementieren können (kleinere Beispiele für PHP und Python werden bereitgestellt). Darüber hinaus ist eine vollständige Implementierung mit Unterstützung für viele Bibliotheken verfügbar: weblablib. weblablib basiert auf dem weitverbreiteten Python-Framework Flask sowie dessen Erweiterungen (Datenbank, Vernetzung, Authentifizierung etc.), wird für die meisten neuen WebLab-Deusto-Labore verwendet und ist speziell darauf ausgerichtet, den Entwicklungsprozess von Remote-Laboren zu beschleunigen.

4.1 Eigenschaften

weblablib bietet folgende Funktionen:

- Flask-Plug-in: Flask ist ein viel genutztes Open-Source-Python-Web-Microframework, das leicht zugänglich und umfassend erweiterbar ist. weblablib lässt sich unkompliziert in Flask integrieren und interagiert auf einfache Weise mit dessen weiteren Komponenten. Die detaillierte Dokumentation der Flask-Umgebung liefert Entwickler*innen wertvolle Informationen zur Einbindung und zum Betrieb des Systems.
- vereinfachtes Modell: Entwickler*innen von Remote-Laboren müssen sich nicht mehr mit Anmeldedaten befassen, da die User bereits in WebLab-Deusto (oder Moodle bzw. einem anderen LMS über LabsLand oder gateway4labs) authentifiziert sind. WebLab-Deusto übermittelt die relevanten Daten an weblablib. Dasselbe gilt für die Terminierung oder Autorisierung: WebLab-Deusto verwaltet Gruppen, User oder den Zugriff auf die Laborkopien, tritt aber nur dann in Aktion, wenn das User-Profil gültig ist.
- User-Informationen: weblablib ist ins Flask-Log-in integriert. Auf diese Weise kann die Anwendung bei jeder Nutzung des Labors einfach auf das User-Profil und den GUID (Globally Unique Identifier) zugreifen. Darüber hinaus lässt sich weblablib in Flask-SQLAlchemy integrieren, um Daten in einer lokalen SQL-Datenbank zu speichern.
- WebSocket: weblablib integriert Flask-SocketIO nativ, was Entwickler*innen von Remote-Laboren dazu befähigt, WebSocket auf sichere Weise zu implementieren (z. B. durch eine entsprechende Authentifizierung). Bei WebSocket handelt es sich um ein modernes HTML5-Protokoll, das es dem Server ermöglicht, Informationen asynchron an die User zu senden – ein Feature, das im Hinblick auf den Betrieb von Remote-Laboren besondere Bedeutung besitzt.

- Multitasking und Aufgabenmanagement: Das System ermöglicht die einfache Einrichtung einer Schnittstelle zum Starten von Hintergrundaufgaben (z. B. Programmierung eines Geräts), die zwar auf die weiteren Funktionen (z. B. User-Identifikation) zugreifen, aber im Batch-Betrieb verarbeitet werden können.
- Internationalisierung (i18n): Das System erkennt, welche Sprache (Englisch, Spanisch ...) WebLab-Deusto anfordert, und unterstützt mittels Flask-Babel bei der Internationalisierung des Remote-Labors.

Darüber hinaus unterstützt das Framework sein eigenes Debugging-System, was Entwickler*innen das Aufrufen oder die Konfigurierung von WebLab-Deusto erspart. Das Ergebnis ist ein unkomplizierter, schneller Entwicklungsprozess. Und sobald das Labor einsatzbereit ist, kann es im laufenden Betrieb in WebLab-Deusto konfiguriert werden.

4.2 Beispiele

Die Dokumentation von weblablib[11] enthält eine Reihe von Anwendungsbeispielen. Abbildung 3 zeigt einen sehr einfachen Code zur Einbindung von weblablib.

```
from flask import Flask, url_for
from weblablib import WebLab, weblab_user, requires_active

app = Flask(__name__)

weblab = WebLab(app)

@weblab.on_start
def on_start(client_data, server_data):
    # ...
    print("Starting user")

@weblab.on_dispose
def on_dispose():
    # ...
    print("Ending user")

@weblab.initial_url
def initial_url():
    return url_for('index')

@app.route('/')
@requires_active
def index():
    return "Hello, {}".format(weblab_user.username)
```

Abbildung 3: weblablib-Musterlabor

11 https://developers.labsland.com/weblablib/

Hierbei wird zum einen festgelegt, welche Funktionen aufgerufen werden sollen, wenn die Nutzung beginnt (z. B.: Wo sollen Aufgaben zur Vorbereitung der Sitzung ausgeführt werden?). Zum anderen wird festgelegt, was zu geschehen hat, wenn die Nutzung endet (z. B. Stopp laufender Motoren und Freigabe verwendeter Ressourcen bei Ablauf der Zugriffszeit oder automatische Überprüfung und Rückführung des Labors in den Ausgangszustand für die nächsten User). Des Weiteren wird gezeigt, wie es möglich ist, Standard-Flask-Webmethoden (*@app.route('/')*) zu verwenden und gleichzeitig Authentifizierungsmechanismen (*@requires_active*) zu unterstützen, die sicherstellen, dass Fremdzugriffe auf diese Website scheitern – es sei denn, WebLab-Deusto hat User umgeleitet und Informationen über sie unter Verwendung des nicht gemanagten HTTP-Protokolls gesendet. Globale Variablen wie *@weblab_user.username* liefern darüber hinaus bei jedem Aufruf auf einfache Weise Daten über aktuelle Nutzer*innen. Das Beispiel in Abbildung 4 zeigt, wie Aufgaben definiert werden können.

```
@weblab.task()
def program_device(contents):
    """ Programs a device. Typically takes 5-10 seconds """
    if weblab_user.time_left < 10:
        raise Exception("Error: user does not have "
                        "enough time to run it!")

    arduino.program("my_file.bin") # In this case
    return len(contents)
# This other code runs it in a different
# process
task = program_device.delay(code)

# The following is a string that you can store in
# Flask session or in weblab_user.data
task.task_id

# a string 'submitted', 'running' or 'failed'/'done' if finished.
task.status

task.submitted  # bool: not yet started by a worker
task.running    # bool: started by a worker, not yet finished
task.done       # bool: finished successfully
task.failed     # bool: finished with error
task.finished   # task.failed or task.done

# These two attributes are None while 'submitted' or 'running'
task.result # the result of the function
task.error # the exception data, if an exception happened

# Join operations
task.join(timeout=5, error_on_timeout=False) # wait 5 seconds
```

Abbildung 4: Verwendung von weblablib-Aufgaben

@weblab.task legt fest, ob eine Funktion wie üblich oder im Hintergrund ausgeführt werden soll. Dabei wird ein Objekt ausgegeben, das definiert, ob die Aufgabe noch läuft, welche Kennung und welchen Status sie hat und ob das System in der Lage ist, gegebenenfalls auf sie zu warten. Ist eine Aufgabe lang, wartet weblablib standardmäßig ihre Beendigung ab, bevor die Ressourcen in den Ausgangszustand zurück-

gesetzt werden. Das dargestellte Beispiel ist sehr einfach und gibt deshalb keine Auskunft darüber, wie die Kommunikation mit der Aufgabe unterstützt wird, um diese zu stoppen oder Informationen über ihren Status zu bekommen. Auch wird nicht deutlich, wie Entwickler*innen innerhalb der Aufgabe Zugang zu Informationen über aktuelle User erhalten (z. B. User-Name, vollständiger Name der Person oder Sprache). Abbildung 5 veranschaulicht, wie WebSocket mit weblablib verwendet werden kann.

```
from weblablib import socket_requires_active

@socketio.on('connect', namespace='/mylab')
@socket_requires_active
def connect_handler():
    emit('board-status', hardware_status(), namespace='/mylab')

@socketio.on('lights', namespace='/mylab')
@socket_requires_active
def lights_event(data):
    switch_light(data['number'] - 1, data['state'])
    emit('board-status', hardware_status(), namespace='/mylab')
```

Abbildung 5: WebSocket-Nutzung mit weblablib

Bestimmte Methoden wie *@socket_requires_active* wurden hier so angepasst, dass sichergestellt ist, dass WebSocket nur von berechtigten Usern mit gültigem Zugriffs-Slot geöffnet werden kann – sowohl bei der Verbindung als auch beim Ausgeben von Daten. Auf diese Weise stellt weblablib bereits direkt nach Beendigung des Zugriffs standardmäßig einen Sicherheitsmechanismus bereit, der verhindert, dass der Client Informationen an die Hardware versendet.

Anhand des Codes in Abbildung 6 wird deutlich, wie das System auch eine native Integration von Datenbanken unter Verwendung der Flask-SQLAlchemy-Bibliothek unterstützt.

```
# Using Flask-SQLAlchemy ( http://flask-sqlalchemy.pocoo.org/ )
from .models import LabUser

@weblab.user_loader
def load_user(username_unique):
    return LabUser.query.filter_by(username_unique=username_unique)

@app.route('/files')
@requires_active
def files():
    user_folder = weblab_user.user.folder
    return jsonify(files=os.listdir(user_folder))
```

Abbildung 6: Nutzung der weblablib-Datenbank

Das globale *weblab_user*-Objekt unterstützt den Aufruf von *user*-Objekten, um intern Zugriff auf ein Datenbankobjekt zu erhalten, das wiederum bestimmte Informationen (z. B. einen Ordner mit Dateien) enthalten könnte.

Die vollständige Struktur dieses API ist in der weblablib-Dokumentation festgehalten.

5 Zusammenfassung und Ausblick auf zukünftige Entwicklungen

Die Entwicklung eines Remote-Labors ist eine komplexe Aufgabe, die aufseiten der Programmierenden erheblichen Aufwand und großes Sachverständnis erfordert. WebLab-Deusto verfolgt in diesem Kontext zwei verschiedene Ansätze, um den unterschiedlichen Entwickler*innen-Typen gerecht zu werden. Die erste Option ist der gemanagte Ansatz mit einer Vielzahl von APIs für verschiedene Programmiersprachen. Der nicht gemanagte Ansatz – die zweite Option – ist zwar grundsätzlich komplizierter, eröffnet erfahrenen Entwickler*innen jedoch auch mehr Handlungsspielraum. Er unterstützt eine Web-Schnittstelle, bietet aber auch weblablib als Open-Source-Framework. Damit steht Entwickler*innen ein einfach zu implementierendes Toolkit zur Verfügung, was den Aufwand zur Integration des Frameworks (Flask für Python und dessen Erweiterungen) erheblich reduziert.

Literaturverzeichnis

Aktan, B.; Bohus, C.; Crowl, L. & Shor, M. (1996). Distance learning applied to control engineering laboratories. *Education, IEEE Transactions on* 39(3), 320–326.

Buitrago, P. A.; Camacho, R.; Pérez, H.- E.; Jaramillo, O.; Villar, A.; Rodríguez-Gil, L. & Orduna, P. (2020). Mobile Arduino Robot Programming Using a Remote Laboratory in UNAD: Pedagogic and Technical Aspects. In *Proceedings of 17th International Conference on Remote Engineering and Virtual Instrumentation (REV2020)*, 26–28 February 2020, University of Georgia, Athens, GA, USA, S. 637–651.

Carisa, B.; Burain, A.; Molly, H. S. & Lawrence, C. (1995). *Running control engineering experiments over the internet*. Oregon State University, Corvallis, OR, USA.

Cedazo, R.; Sanchez, F.; Sebastian, J.; Martinez, A.; Pinazo, A.; Barros, B. & Read, T. (2006). Ciclope chemical: a remote laboratory to control a spectrograph. *Advances in Control Education* 39(6), 517–522.

Coble, A.; Smallbone, A.; Bhave, A.; Watson, R.; Braumann, A. & Kraft, M. (2010). Delivering authentic experiences for engineering students and professionals through e-labs. *Education Engineering* (EDUCON), 2010 IEEE, 1085–1090.

Del Alamo, J.; Brooks, L.; McLean, C.; Hardison, J.; Mishuris, G.; Chang, V. & Hui, L. (2002). The MIT microelectronics weblab: A web-enabled remote laboratory for microelectronic device characterization. In: *World Congress on Networked Learning in a Global Environment*, Berlin, Germany.

Gillet, D.; Latchman, H.; Salzmann, C. & Crisalle, O. (2001). Hands-on laboratory experiments in flexible and distance learning. *Journal of Engineering Education* 90(2), 187–191.

Gillet, D.; de Jong, T.; Sotirou, S. & Salzmann, C. (2013). Personalised learning spaces and federated online labs for stem education at school. In: *Global Engineering Education Conference (EDUCON)*, 2013 IEEE, S. 769–773.

Gomes, L. & Bogosyan, S. (2009). Current trends in remote laboratories. *Industrial Electronics,* IEEE Transactions on 56(12), 4744–4756.

Gravier, C.; Fayolle, J.; Bayard, B.; Ates, M. & Lardon, J. (2008). State of the art about remote laboratories paradigms-foundations of ongoing mutations. *International Journal of Online and Biomedical Engineering (iJOE)* 4(1), 19–25.

Gustavsson, I.; Zackrisson, J.; Håkansson, L., Claesson; I. & Lagö, T. (2007). The VISIR project – an open source software initiative for distributed online laboratories. In: *Proceedings of the REV 2007* Conference, Porto, Portugal.

Hardison, J.; DeLong, K.; Bailey, P. & Harward, V. (2008). Deploying interactive remote labs using the ilab shared architecture. In: *Frontiers in Education Conference, 2008.* FIE 2008. 38th Annual. pp. S2A–1. IEEE.

Henry, J. (1996). Running laboratory experiments via the world wide web. In: *ASEE Annual Conference*.

Jong, T. de; Linn, M. C. & Zacharia, Z. C. (2013). Physical and virtual laboratories in science and engineering education. *Science 340*(6130), 305–308.

Kwinana, P. M.; Nomnga, P.; Rani, M. & Lekala, M. L. (2020). Real laboratories available online: Establishment of ReVEL as a conceptual framework for implementing remote experimentation in South African Higher Education Institutions and rural-based schools – A case study at the University of Fort Hare. In: *Proceedings of 17th International Conference on Remote Engineering and Virtual Instrumentation* (REV2020), 26–28 February 2020, University of Georgia, Athens, GA, USA, S. 402–414.

Lowe, D.; Machet, T. & Kostulski, T. (2012). Uts remote labs, labshare, and the sahara architecture. In: García-Zubía, J. & Alves, G. R. (Hrsg.), *Using Remote Labs in Education: Two Little Ducks in Remote Experimentation* (Vol. 8) (S. 403–424). Universidad do Deusto, Bilbao, Spain.

Nedic, Z.; Machotka, J. & Nafalski, A. (2008). Remote laboratory netlab for effective interaction with real equipment over the internet. In: *Human System Interactions*, 2008 Conference on, S. 846–851. IEEE.

Orduña, P. (2013). Transitive and Scalable Federation Model for Remote Laboratories. Ph.D. thesis, Universidad de Deusto, Bilbao, Spain. Verfügbar unter https://more lab. deusto.es/people/members/pablo-orduna/phd_dissertation/ [09.10.2020].

Orduña, P.; Almeida, A.; Ros, S.; López-de-Ipiña, D. & García-Zubia, J. (2014). Leveraging Non-explicit Social Communities for Learning Analytics in Mobile Remote Laboratories. *Journal of Universal Computer Science* 20(15), 2043–2053.

Orduña, P.; Bailey, P.; DeLong, K.; López-de-Ipiña, D. & García-Zubia, J. (2014). Towards federated interoperable bridges for sharing educational remote laboratories. *Computers in Human Behavior* 30, 389–395. Verfügbar unter http://www.sciencedirect. com/science/article/pii/S0747563213001416. [09.10.2020].

Orduña, P.; Garbi Zutin, D.; Govaerts, S.; Lequerica Zorrozua, I.; Bailey, P. H.; Sancristobal, E.; Salzmann, C.; Rodriguez-Gil, L.; DeLong, K.; Gillet, D. et al. (2015). An extensible architecture for the integration of remote and virtual laboratories in public learning tools. *Tecnologias del Aprendizaje, IEEE Revista Iberoamericana de* 10(4), 223–233.

Richter, T.; Boehringer, D. & Jeschke, S. (2011). Lila: A european project on networked experiments. *Automation, Communication and Cybernetics in Science and Engineering* 2009/2010, 307–317.

Šafarič, R.; Truntič, M.; Hercog, D. & Pačnik, G. (2005). Control and robotics remote laboratory for engineering education. *International Journal of Online Engineering* (iJOE) 1(1), 1–8.

Schauer, F.; Krbecek, M.; Beno, P.; Gerza, M.; Palka, L.; Spilakov, P. & Tkac, L. (2016). Remlabnet iii – federated remote laboratory management system for university and secondary schools. In: *2016 13th International Conference on Remote Engineering and Virtual Instrumentation (REV).*, S. 238–241. IEEE (2016).

Torres, F.; Candelas, F.; Puente, S.; Pomares, J.; Gil, P. & Ortiz, F. (2006). Experiences with virtual environment and remote laboratory for teaching and learning robotics at the university of alicante. *International Journal of Engineering Education* 22(4), 766–776.

Zappatore, M.; Longo, A. & Bochicchio, M. A. (2016). Enabling MOOL in acoustics by mobile crowd-sensing paradigm. In: *2016 IEEE Global Engineering Education Conference (EDUCON)*. S. 733–740. IEEE (2016).

Abbildungsverzeichnis