

1. Unisync - Introduction

Unisync enables the automated creation of messenger chat groups that correspond to the course rooms in ILIAS. The lightweight solution, which can also be adapted to other LMSs, was developed as part of a sub-project of the InduKo project funded by the Stiftung Innovation in der Hochschullehre.

One constraint that had to be taken into account during development was the fact that there was no system access to ILIAS, which is why it was not possible to transfer data via system interfaces. Without a direct connection to ILIAS, the use of a web scraper was the only way to read course data from the system and then transfer it to the Matrix system. The great advantage of this lightweight solution is that it does not require in-depth system integration or adaptations to ILIAS or other learning management systems. This not only makes it easier to adapt to other environments and learning management systems, it also significantly reduces the security risk, as no intervention in existing systems is required and their integrity and proven security mechanisms are fully preserved.

Unisync does not require any additional app installation for course synchronization. A link, which can be stored in Ilias, for example, or as a bookmark in the browser, enables teachers to automatically transfer their Ilias courses to Messenger chat rooms. After logging into the LMS, the course data is then transferred to the Matrix system, where chat groups are automatically created. The course participants are assigned to the groups and invitation links are sent out. The chat groups are then visible in Matrix-based messengers such as Element or FluffyChat, depending on the university's preferences.

This solution is flexible, independent of specific front-ends or system structures, reduces maintenance costs and increases security. Users do not need to install any additional software and the solution can be adapted to the needs of different universities. Unisync was developed as part of a subproject within the InduKo project (Innovation through Collaboration), which was funded by Stiftung Innovation in der Hochschullehre.

2. Unisync - core aspects and architecture

Unisync is an efficient and flexible synchronization solution for the automated creation of matrix chat rooms for ILIAS courses. To achieve these goals, an architecture was developed in which a matrix server acts as an API endpoint that enables room creation, synchronization and invitation of users. This process eliminates manual intervention and ensures that all students enrolled in Ilias courses are included in the appropriate communication rooms. OpenID Connect is used for authentication, supplemented by two-factor authentication (2FA) for additional security. All communication content is protected with Matrix's end-to-end encryption. To make use as intuitive as possible, special emphasis was placed on a smooth user experience. A minimalist, intuitive interface facilitates the login process and synchronization of courses.

The architecture is based on three main components: the front end, the back end and the matrix server. These work closely together to enable the automatic creation and management of communication rooms that correspond to the course rooms in ILIAS. The following section describes how the system works in detail.

The frontend provides the interface for users to enter login details and monitor the synchronization process. It was developed using technologies such as HTML, CSS, JavaScript and Bootstrap to ensure a user-friendly and responsive interface.

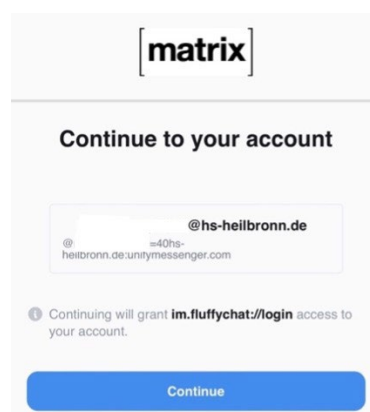
Users start the process by entering their ILIAS access data in the input mask of the Unisync system. If required, the system also supports two-factor authentication (2FA) to increase security. They can also use the user interface to track the progress of the synchronization in real time and view details about the matrix rooms, such as room ID and participant lists.

Communication between the front end and the back end takes place via AJAX (Asynchronous JavaScript and XML). This technology makes it possible to update data in real time without the need for a full page refresh. All data exchanged between the frontend and the backend is HTTPS encrypted to ensure the confidentiality of data transmission.

The backend is the central component for data processing and synchronization. It was developed with FastAPI, a powerful and modern web framework for Python that enables fast and efficient API communication. One of the main tasks of the backend is to authenticate users to the ILIAS system. This process is based on the OpenID Connect protocol (OIDC).

After successful authentication, an automated web scraper is started that works with the help of Playwright and BeautifulSoup4 technologies. This scraper navigates independently through the ILIAS system and retrieves the list of assigned courses and participants. All relevant data, including course titles, descriptions and participant lists, are extracted. This information is crucial for later synchronization with the Matrix system.

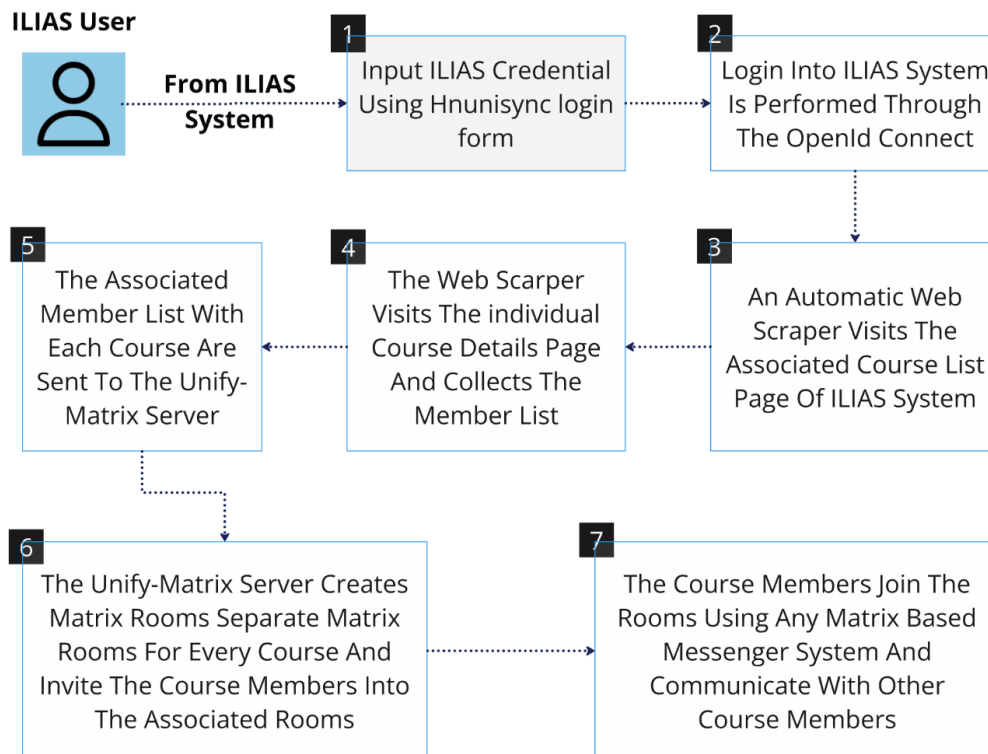
The extracted data is analyzed and processed in the backend. In order for the user assignment to work smoothly, it is necessary that the user identities in ILIAS and in the matrix server clearly match each other, which can be ensured either by creating the users in the matrix server in such a way that they match the ILIAS users or by a translation in the backend: i.e. if there are differences in the conventions of the two systems, the backend can translate the ILIAS user information into the corresponding matrix users, since the naming convention according to which the users were created in the matrix server via OpenID is known.



This screen shows the matrix user as it was created by the connection to OpenID. The users must either be identical or, as in this case, a corresponding mapping must be carried out.

This mapping is required if a direct connection or integration with system access at the university is not possible for security reasons. The backend then takes over communication with the Matrix Synapse server. The Matrix-Nio library is used here, which enables instructions for creating new chat rooms to be transmitted and participants to be assigned to the corresponding rooms. This ensures that only correctly assigned users can be included in the Matrix rooms. This procedure guarantees a functioning solution within the given restrictions.

Hnunisync and Unify-Matrix Process flow



The Matrix Synapse server acts as a central platform for the administration of communication rooms; one of its central tasks is the creation of chat rooms for each course in ILIAS. The structure of the created rooms reflects the course organization in ILIAS, so that a clear assignment between courses and communication rooms is guaranteed.

Another important aspect is the automatic management of participants. Based on the data provided by the backend, the Synapse server invites the relevant participants to the corresponding chat rooms. This process is enabled by Matrix's API-supported functions and is initiated entirely by the backend. The Synapse server ensures secure communication through the use of end-to-end encryption. This technology protects all messages and data from unauthorized access and ensures the confidentiality and integrity of communication within the rooms. As a result, exchanges between teachers and students remain completely private and secure.

Once the rooms have been created, users receive invitations that allow them to join the communication rooms. These rooms can be used with any Matrix-based messenger application, including Element, FluffyChat and others.

Unisync - User - Flow: From login to synchronization and communication

Unisync offers a secure login process based on the university access data. The users who carry out this synchronization are teaching staff. The authentication process begins with users entering their university credentials. This entry takes place in the Unisync user interface, which can be seen in the following illustration.

ILIAS-Kurse mit Matrix-Räumen synchronisieren

Mit HHN-Account anmelden

Benutzername oder E-Mail:

Passwort:

After entering the access data, the user's identity is verified via the OpenID Connect authentication process, which ensures that only authorized persons have access to the platform. After users have entered their HHN access data, an additional step is required: the entry of a one-time verification code (OTP). This code is generated by an authentication app such as Google Authenticator.

The following illustration shows the user interface in which the one-time verification code is entered.

ILIAS-Kurse mit Matrix-Räumen synchronisieren

Eingabe eines Verifizierungscode aus der Authenticator-Anwendung.

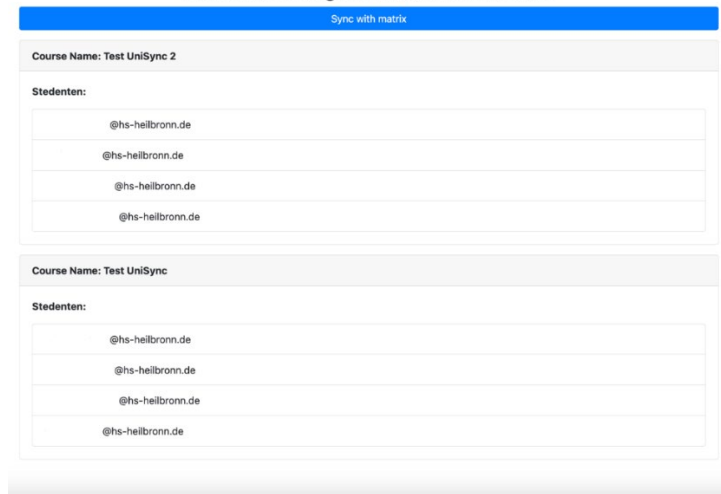
One-time code:

After successful authentication, Unisync uses the web scraping mechanism to automatically collect information about the courses enrolled in the ILIAS system and the associated participants. This automated process extracts details such as the course names as well as the complete participant lists.

The collected information is presented in a clear user interface that allows users to synchronize their courses with the corresponding Matrix rooms. This is done by simply clicking on the "Sync with Matrix" button.

ILIAS-Kurse mit Matrix-Räumen synchronisieren

Kursliste mit eingeschriebenen Studenten



The screenshot shows two course lists. The first list is titled 'Course Name: Test UniSync 2' and contains four student email addresses: @hs-heilbronn.de. The second list is titled 'Course Name: Test UniSync' and also contains four student email addresses: @hs-heilbronn.de. The interface has a blue header bar with the text 'Sync with matrix'.

To complete the synchronization process, users log in to the Unify Matrix server with their Matrix user ID and password. This secure login ensures that Unisync can connect to Matrix and correctly add students to the appropriate chat rooms.

Matrix Anmeldung



Bitte melden Sie sich bei Matrix an, um den Synchronisierungsprozess abzuschließen.

Matrix Benutzer ID:

Matrix Passwort:



Matrix Anmelden

Unisync now automatically creates matrix rooms named after the respective courses and invites all enrolled students to join the corresponding rooms. A confirmation page provides teachers with an overview of the synchronization process. This includes the matrix room ID and a list of successfully added students.

The view for the user is determined by the selection of the preferred matrix messenger. Here, for example, is a screenshot of the open source messenger FluffyChat with the corresponding transparent invitation notice; after confirmation by the user, the user automatically joins the course



Brief description of the technology stack used

The stack used is divided into two main components: the web application and the matrix server.

The Unisync web application uses several frameworks and libraries to implement user authentication, course synchronization, web scraping and communication with the matrix server. The central technologies include:

- Python FastAPI is used for the creation of APIs.
- Uvicorn Serves as an ASGI server for the execution of FastAPI applications.
- Pydantic takes over the validation and processing of data.
- Jinja2 is used for the dynamic creation of HTML content using templates.
- Gunicorn enables the efficient execution of Python web applications.

The following tools are used for the automation of web scraping processes:

- Playwright automates browser interactions to retrieve course data from the ILIAS system.
- BeautifulSoup4 analyzes HTML documents and extracts structured information from the ILIAS system.

Interaction with APIs and support for asynchronous processes is made possible by the following technologies:

- Matrix-Nio, this client library is used to interact with the Matrix protocol and supports the management of communication rooms and participants.
- Quart, a framework that enables asynchronous processes and is compatible with Python asyncio.
- Requests and Httpx are used to carry out HTTP requests and communicate with APIs.

For the interaction with the matrix protocol are used:

- Matrix-Nio, this client library enables interaction with Matrix servers and functions such as room creation and participant management.
- Matrix-Synapse is the Matrix home server used to manage and host Matrix rooms.

The following technologies are used on the backend side:

- Quart enables the processing of asynchronous backend processes.
- Request is used to make HTTP requests to external services.

Login process: Current workaround and possible solution

After a server change, our newly installed Matrix server, including a new domain, no longer had access to the OpenID Connect APIs of our university compared to the previous version. As a result, it is not possible to use the standard, token-based authentication process.

To get around this problem, the backend uses the automation tool Playwright. This simulates user interactions and takes over the entry of the access data and the one-time code (OTP) on behalf of the user. It navigates through the login process provided by the identity provider. Although this solution is functional, it deviates from the standards of the OIDC protocol and therefore entails security and compliance risks. The lack of direct access to the university OpenID Connect API also increases the complexity of the system. Maintenance of the automation scripts proves to be time-consuming and error-prone, as changes to the identity provider's login workflow can interrupt the automation. This requires continuous monitoring and regular updates to maintain functionality.

This is a workaround, as direct access to the OpenID Connect API provided by the university was no longer made available after the server change. However, this approach is not recommended as a best practice solution. Instead, the standardized use of OpenID Connect should be chosen. Nevertheless, for the sake of completeness, the weaknesses of the chosen workaround for the login process are briefly described below.

Entering a username, password and OTP directly into the Unisync login screen leads to potential security risks such as unauthorized access, man-in-the-middle attacks and potential exposure of sensitive data in storage. In addition, the end-to-end encryption provided by OIDC is missing, which could put user data at risk in the event of a security incident in the backend. Bypassing the consent page of the IdP compromises transparency and consent. Without ID or access tokens, important security features such as token verification, expiration and revocation are missing. Users are not sufficiently informed that their login data is processed directly by Unisync. The dependency on Playwright increases the maintenance effort, as changes to the IdP login page require frequent updates to the automation scripts. Resource-intensive browser automation puts a strain on system performance and can fail due to network delays or security measures such as captchas, which interrupts the workflow

To avoid the described vulnerabilities of the workaround, it is recommended to integrate Unisync directly with the OIDC provider of the respective university. This includes the configuration of a client

ID, a client secret and secure redirect URIs. This direct integration enables the use of OIDC standards, simplifies authentication processes and increases security. In future, users will be redirected to the IdP login page and, after successful authentication, the IdP will provide an ID token for identity verification and an access token for authorizing data access. Authentication is implemented by redirecting to the IdP login page and validating the tokens provided in Unisync. The user interface remains unchanged, while token-based workflows increase security and reliability.

Changes to the server configuration are required for OIDC integration. The CORS settings must be adjusted to allow requests from the IdP and firewall rules must be updated to enable secure communication. In addition, the Unisync domain must be registered as trustworthy in the IdP settings.

The integration of a standardized OIDC workflow eliminates security risks, ensures adherence to compliance standards, reduces system complexity and improves the stability and scalability of the system.

Extensive testing beyond the development team and a field test were not conducted during the project.