**Project Report**

in the module

**Architecture & Integration**

Mr. Sebastian Speck

Project Group

Nour Batniji 1507535

Saida Covrk 1509852

Mariam Zadran 1518535

**Introduction**

As part of the project in the module Architecture and Integration, our group implemented an integration scenario involving three applications at the Frankfurt University of Applied Sciences. The objective was to automate the transfer of student data from the central HIS system to the two dependent applications, Peregos and WyseFlow, using a middleware-based architecture. The project included both an architectural modeling and a prototype implementation using an integration middleware.

In the previous setup, student data, such as name, IDs, enrolled study programs and earned credit points, was maintained centrally in HIS. This data had to be manually re-entered into Peregos and WyseFlow, leading to increased workload and a higher risk of inpur errors.

In response to these challenges, the head oft he Examination Office requested the development of an automated solution that should distribute student data from HIS to Peregos and WyseFlow.

In the following, our group presents the architecture and the prototypical implementation we developed for this purpose, describe the involved systems and their interactions and explain how the data flow was implemented using RabbitMQ as the messaging middleware.

**Project Objective**

The main objective was to implement an integration middleware that automates the flow of student data from HIS to both Peregos and WyseFlow. The system should:

* Ensure data consistency
* Reduce manual effort and errors
* Be modular and easily extendable

**Architecture**

The modeled architecture illustrates the data flow involved in the management of student examinations and academic processes at a university. The core actors in this system are the Student and the Examinitation office. Both of them interact with multiple application services to perform key academic and administrative tasks.

At the core of the system lies the HIS application, which acts as a data producer. It enables students to register for and withdraw from examinations, view information about their registered or deregistered exams and track their accumulated credit points. All these actions are processed trough the „examination management“ functiom within HIS. Once student-related data is updated, HIS publishes this information for further administrative use. The HIS System operates on a dedicated server, which is compatible with Windows, Linux and macOS ensuring flexibility and broad accessibility. All data is stored securely within this environment.

To facilitate inter-system communication, the published student data is transmitted trough RabbitMQ in JSON format. The message queue acts as a middleware, that routes the data to the two primary consumer systems: Peregos and WyseFlow. Both consume the data independently and process it according to their specific functions.

Two systems serve as data consumers: Peregos and Wyseflow.

Peregos is responsible for handling administrative processes related to student data. It includes the functions student data management and request management, which are used to create and process student requests. These processes are initiated by students and managed by the Examinitaion Office. Peregos receives the student data through RabbitMQ stores relevant information in its own database and operates on a server environment that supports Windows, Linux and macOS.

WyseFlow is responsible for the thesis application management. Once student data is received, the system processes applications trough the functions student data management and thesis management. This allows students to apply for their thesis and enables the Examination Office to manage the entire application process. The WyseFlow server also supports Windows, Linux and macOS and maintains all relevant information in its own database.

In conclusion, the architecture represents a modular and scalable system that promotes separation of concerns. HIS acts as the authoritative data source, broadcasting structured student data to Peregos and WyseFlow trough RabbitMQ. This setup allows each system to function independently, ensuring that updates or changes in one component do not negatively affect the others.

**Implementation**

System Components

1. *His – Sender Interface*

* Provides a GUI for entering student data including name, matriculation number, and study programs.
* Implements data validation for name format, ID digits, and allowed study programs.
* Sends student data to Peregos and WyseFlow via RabbitMQ (student\_exchange) using the routing keys:
* Peregos.info
* Wyseflow.info
* Includes:
* Ping mechanism to ensure connection availability of receivers. HIS sends a ping to WyseFlow and Peregos every five seconds and they answer with a pong if they are online.
* Error listener for receiving validation feedback from target systems.
* Student overview dialog to reviw sent data that is stored locally (students.json).

1. *Peregos – Receiver Interface*

* Listens to peregos.info and peregos.ping routing keys.
* Accepts and validates student data received from HIS:
* Only predefined programs are accepted (Computer Science, Economics, AI).
* Logs data and stores it in a local file (received\_students\_peregos.json).
* Sends validition errors back to HIS via the routing key his.error.
* Responds to ping messages with a his.pong reply.

1. *WyseFlow – Receiver Interface*

* Listens to wyseflow.info and wyseflow.ping routing keys.
* Similar to Peregos in behavior and validation.
* Stores accepted student data in received\_students\_wyseflow.json.
* Responds to ping requests and reports unkown programs trough his.error.

Middleware Configuration

The core of the communication relies on RabbitMQ:

* Each system declares and binds queues to specific routing keys:
* His.pong, his error (HIS listens)
* Peregos.info, peregos.ping (Peregos listens)
* Wyseflow.info, wyseflow.ping (WyseFlow listens)
* All messages are JSON encoded and passed via pika (Python RabbitMQ client).

Communication Flow

1. Initialization:

* HIS sends ping requests to Peregos and WyseFlow.
* Each system responds with a pong message to confirm availability.

1. Data Transmission:

* Upon successful connectivity, HIS sends validated student data to both Peregos and WyseFlow.
* Both systems receive the data, validate it, and store it locally.

1. Error Handling:

* If either receiver identifies unknown programs, an error message is sent back to HIS.
* HIS displays the error via a GUI message box.

Testing

* Tested sending valid and invalid study programs.
* Confirmed that Peregos and WyseFlow both only accept predefined programs.
* Verified that connection status and pong responses work correctly.
* Ensured that error messages are routed back and displayed in HIS.

**Errors & Mitigation Strategies**

During the development and testing oft he integration scenario, several potential errors were identified that could impact the data consistency and system communication. To ensure the reliability oft he automated data exchange, mitigation strategies were implemented at both the sending and receiving ends.

The following table outlines the key error sources and the corresponding measures taken to prevent or handle them effectively.

|  |  |  |
| --- | --- | --- |
| **Error** | **Description** | **Solution** |
| **Message lock** |  |  |
| **False Data** |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Conclusion**

The developed integration architecture successfully addresses the challenges previously caused by manual data entry between HIS, Peregos and WyseFlow. By introducing RabbitMQ as a central messaging middleware, student data can now be transmitted automatically, reliably and fast. Each systm processes the received data independently based on its specific requirements, which ensures a clear seperation of concerns and improves maintainability.

The modular design allows future extensions such as the integration of additional systems. Overall, the project demonstrates how middleware-based integration can significantly improve data consistency, reduce administtrative workload and create a scalable foundation for further digitalization at the university.

**Implementation Results (kurze beschreibung zu allen? Ich kann keine bilder bei gpt hochladen der kann schnell ein satz zu jedem schreiben)**







