Master’s Thesis
Niklas Meißner

Interdisciplinary Composition of E-Learning Platforms based on Reusable Low-Code Adapters

Prototype Documentation

Contents

Overview

This section provides an overview of the prototype.

Current state

The prototype was designed according to the microservice architecture and contains an Angular frontend and a Java Spring Boot backend, which communicates with each other using a REST API.
In the background acts a MySQL server, which is installed locally and provides databases for all services.

The following table shows the services, including the addresses at which they can be reached:

Part of the Application	Services	Addresses
Frontend	Frontend	http://localhost:4200
Backend	Adapter Service	http://localhost:8081
	Course Service	http://localhost:8084
	Dashboard Service	http://localhost:8085
Low-Code Adapter	Course Timeline Adapter	http://localhost:8091
	Flashcards Adapter	http://localhost:8093
	Gamification Adapter	http://localhost:8095
	Knowledge Progress Adapter	http://localhost:8097
	Quiz Adapter	http://localhost:8099
	Video Adapter	http://localhost:8101
	Matching Exercise Adapter	http://localhost:8103
MySQL Server	-	http://localhost:3306

Docker & Kubernetes

The application is Docker and Kubernetes ready. The repositories already contain the required Dockerfiles. For reasons of time and effort of the prototype, a direct implementation of the application in Kubernetes is waived here and could be implemented as the next step. In addition, a gateway must be set up that forwards requests and communication from services directly to the correct service. For this, the currently locally installed MySQL database server and services need to be put into a pod and connectivity between them needs to be established. Once all the services and adapters are in Docker containers, the external IP addresses need to be stored in the gateway so that the requests from the services are forwarded to the right service. Thus, changing addresses would not pose any problems.

Setup Process

This section describes how to set up the prototype on the local machine.

Prerequisites

- Node.js 16.13.x and Git Installation
- MySQL 8.x Installation
- Java 11 (openjdk) and Maven 3.8.x Installation
- Angular 13.x Installation
- Make sure all ports for the services are not blocked

Get started

Step 1 – Clone Git repository
Clone git repository of source code: https://github.com/Nimeggis/MA-prototype.git

The repository contains the following folder structure:

- Adapter
    - courseAdapter
    - flashcardsAdapter
    - gamificationAdapter
    - knowledgeAdapter
    - matchingAdapter
    - quizAdapter
    - videoAdapter

- Backend
    - adapterService
    - courseService
    - dashboardService

- Database

- Frontend

Step 2 – Setup DB
Run a local MySQL server on Port 3306 and import the data.
! The microservices use the user: *root* with the pw: *password*. Either create the user for the database server this way, or adjust the application configuration of the microservices accordingly. !
Import the following SQL: *Database > createPrototypeDB.sql*
The databases as well as the tables are automatically generated and filled with the necessary information.

Step 3 – Start Backend
Before running the individual microservices of the backend, a *npm install* of each service should be performed to bring the node modules up to date. After that, the three services (adapterService, courseService and dashboardService) can be started as Java Spring Boot applications.

Step 4 – Start Low-Code Adapter
Before running the individual microservices of the adapters, a *npm install* of each service should be performed to bring the node modules up to date. The low-code adapters consist of two services, the businessLogicService and the configurationService. This prototype includes only the configurationService of each adapter, so no businessLogicService needs to be started. Thus, the configurationService must be started by each required adapter.

Step 4 – Start Frontend
To start the frontend only one command has to be executed in the terminal in the frontend folder: *ng serve --open*
Afterwards the frontend including the login mask should open directly.

Application Architecture

Prototype Overview

In the following, the functions of the individual services of the prototype are explained:

Frontend

User interface for base platform with the following functions:
- Receive user input and forward to backend
- Receive and visualize information from backend

Adapter Service

Service to display available low-code adapters as well as possible compatibilities with other adapters in a course. The adapter service has the following functions:
- Listing of the available low-code adapters
- Manages possible compatibilities with other adapters
- Manages linking between adapters
- Manages usage of respective adapter

Course Service

Service to create and maintain courses. Course service has the following functions:
- Creation of courses and related information
- Manages installed low-code adapters in course

Dashboard Service

Service creates a customized dashboard depending on course, installed low-code adapters and their configuration. The dashboard service has the following functions:
- Creation of a landing page for students depending on installed low-code adapters and learning progress
- Composition of various functionalities within one course

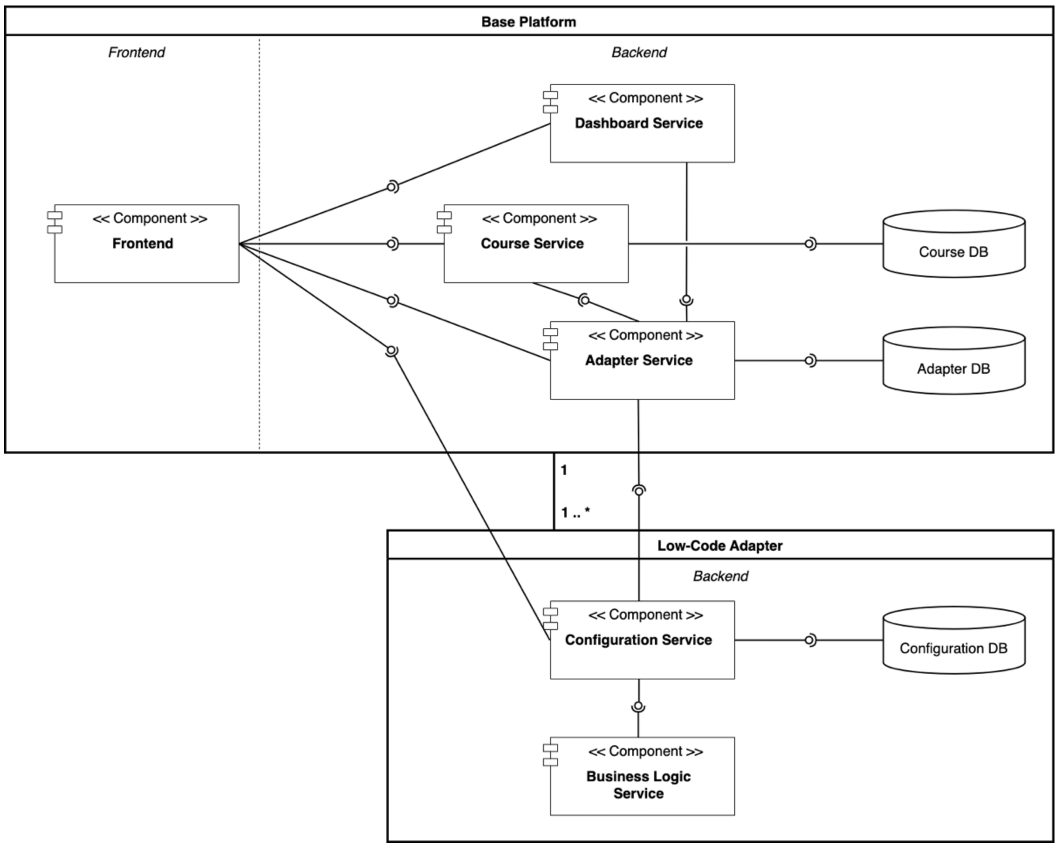
Low-Code Adapter

Configuration Service
Service for configuring the low-code adapter. The configuration service has the following functionality:
- Configuration management of the adapter
- Adjustment of the configuration by authorized users
- Enable dashboard widget for students

Business Logic Service
! This service can be divided and outsourced into different, multiple services depending on the development of the low-code adapter, but this one represents the service for the business logic of the adapters !
The business logic service has no functionality in this prototype so far.

Implementation View

The Implementation view diagram shows the current state of the prototype:



REST APIs

Adapter Service

Address & Parameter	REST	Description
/getAll	GET	Get all adapters from DB and return as list
/getIDs/{adapterName}	GET	Get IDs of adapters from DB and return IDs as CSV (String)
/getAdapter/{adapterID}	GET	Get information of adapter from DB by ID and return as AdapterModel

/manageLink/{courseId}/{adapterSend}/{addOrDel}/{adapterRec}	PUT	link two adapters with each other; one adapter wants to connect to another, sends a request and this function forwards the request to the respective adapter and connect the adapter also the other way

Course Service

Address & Parameter	REST	Description
/getAll	GET	Get all courses from DB and return as list
/getCourse/{courseID}	GET	Get course information by course ID
/getInstalledAdapter/{courseID}	GET	Get installed adapter in course
/createCourse (Course info in request body)	POST	Create course with form values in request body
/deleteCourse/{courseID}	DELETE	Delete course by course ID
/updateCourse/{ ourseID}/{courseAdapter}	PUT	Update installed adapters when adapters are installed/uninstalled

Dashboard Service

Address & Parameter	REST	Description
/getDashAdapter/{courseID}	GET	Get adapter of course which are enabled as dashboard widget

Low-Code Adapter

All low-code adapters have similar REST APIs to address the adapter configuration.

Address & Parameter	REST	Description
/getConfig/{courseID}	GET	Get adapter configuration for specific course
/createConfig/{courseID}	POST	Create initial adapter configuration with default values
/deleteConfig/{courseID}	DELETE	Delete adapter configuration
/setAdapter/{courseID}/{adapterIDs}	PUT	Receive a request of another adapter to link and add it to linked adapter
/updateAdapter/{courseID}	PUT	Update adapter configuration with new values
/addAdapter/{courseID}/{adapterID}	PUT	Link another adapter with adapter
/delAdapter/{courseID}/{adapterID}	PUT	Unlink another adapter from adapter

Dependencies

- spring-boot-starter-parent (2.7.0)
- spring-boot-starter-webflux
- spring-boot-starter-data-jpa
- spring-boot-starter-web
- spring-boot-starter-test
- mysql-connector-java
- lombok