МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ЛИПЕЦКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Институт компьютерных наук

Кафедра автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА № 4

по дисциплине «Архитектура программных систем»

Интеграция информационных систем на основе сервисов обмена сообщениями

Студенты АС-21-1 _____ Мосякин И.И.

(подпись, дата) Станиславчук С.М.

Руководитель _____ Алексеев В.А.

(подпись, дата)

Липецк 2025 г.

Задание кафедры

Реализовать информационное взаимодействие двух информационных систем посредством использования шины обмена сообщениями (программное обеспечение RabbitMQ, ActiveMQ, Kafka, 1С:Шина и др.; сервисы Azure Service Bus, Amazon Simple Queue Service (SQS), Google Firebase Cloud Messaging или др.).

Лабораторная работа выполняется двумя студентами, каждый отвечает за реализацию своей информационной системы, совместно студенты разрабатывают интеграционное решение. Информационные системы могут быть реальными проектами, ранее разработанными в рамках других дисциплин, или условными прототипами, реализованными для целей данной лабораторной работы.

Цель работы

Изучить задачи интеграции информационных систем, стили и шаблоны реализации интеграции, освоить применение сервисов шины обмена сообщениями для реализации взаимодействия информационных систем.

1 Описание интеграционной задачи

1.1 Характеристика ИС № 1

Целью АИС является автоматизация и улучшение управления учебным процессом в высшем учебном заведении с целью повышения его эффективности и обеспечения более качественного образования для студентов. АИС предназначена для сбора, хранения и предоставления информации, необходимой для всех участников учебного процесса, включая студентов, преподавателей, а также администраторов данной системы; для автоматизации административных задач, связанных с учебным процессом.

Перечень решаемых задач:

Учет студентов: АИС должна позволять учреждению вести учет всех студентов, включая личные данные, контактную информацию, учетные записи и другие сведения.

Учет преподавателей: Система должна поддерживать информацию о преподавателях, их квалификации, учебных нагрузках и контактных данных.

Расписание занятий: АИС должна автоматизировать процесс создания и управления расписанием практик, лекций и других учебных мероприятий.

Учет успеваемости и оценок: Система должна позволять вводить и отслеживать оценки, успеваемость студентов и предоставлять студентам и преподавателям доступ к этой информации.

Учет учебных предметов и программ: АИС должна содержать информацию о предметах, учебных программах и учебных планах.

1.2 Характеристика ИС № 2

Программа, отправляющая сообщение на шину RabbitMQ (порт 5672) в формате CSV.

1.3 Задача интеграции

Задачей интеграции является передача информации о группе студента, который получил оценку.

2 Проектирование интеграционного решения

2.1 Преимущества «обмена сообщениями» для интеграции

- Слабые связи между системами, участвующими в интеграции. Более того, в правильно построенной интеграционной модели система вообще ничего не знает о других участниках интеграционного ландшафта. Вся работа сводится к передаче сообщений в сервисную шину и прием сообщений от шины. Этим достигается самый высокий уровень гибкости и масштабируемости относительно всех ранее рассмотренных систем.
- Возможности для трансформации данных. Позволяют интегрировать приложения, рассчитанные на различные форматы данных, без необходимости их доработок. Это помогает снизить затраты на обработку данных системами (данные отправляются один раз в формате системы-источника и принимаются системами-потребителями в своих «родных» форматах), а также задействовать в интеграции системы, доработка которых невозможна или крайне нежелательна по тем или иным причинам. Причем затраты на трансформацию данных не ложатся на интегрируемые системы.

- Маршрутизация данных. Один из важнейших механизмов сервисной шины, позволяющий резко снизить зависимость и связанность участников интеграционных процессов. При наличии механизма маршрутизации система-источник может просто однократно передать сообщение в шину. Ей не требуются знания о том, кто должен получить это сообщение, готов ли он к приему сообщения и т.д. Сообщение будет доставлено всем потребителям в соответствии с текущим маршрутом.
- Соответственно, при масштабировании схемы нам также не требуется вносить изменения во все системы. Достаточно внести изменения в маршрут, добавив или удалив потребителя данных. Это также позволяет выполнять доставку сообщений согласно определенным условиям. Причем сами системы не участвуют в определении условий прохождения маршрута, а значит это поведение может быть легко модифицировано без необходимости вносить изменения в системы.
- Гарантированная доставка данных. Этот механизм сервисной шины предприятия существенно упрощает схемы доставки данных на каналах с низкой стабильностью, снимая нагрузку с систем-источников. Им не приходится реализовывать механизмы для проверки наличия канала связи и промежуточные хранилища для сообщений на время отсутствия канала доставки. Также этот механизм позволяет снизить алгоритмическую нагрузку по реализации механизма квитирования доставки. Этот функционал реализован на уровне интеграционных механизмов сервисной шины.
- Обеспечение безопасности при передаче данных. Ни для кого не секрет, что во многих случаях утечка конфиденциальных данных происходит именно при их передаче. Шины обеспечивают шифрование передаваемых данных, а также поддерживают

защищенные сетевые соединения.

- Централизованное управление интеграционной схемой является важным компонентом любого интеграционного ландшафта. Такой подход сильно снижает накладные расходы на первичную настройку, масштабирование и поддержание работоспособности схемы в целом. Также это позволяет сконцентрировать необходимые компетенции в одном месте, не распыляя их по интегрируемым системам.
- Диагностика состояния. Важной особенностью использования специализированных сервисных шин являются механизмы диагностики. Использование этих механизмов позволяет выявить проблемы, связанные как с передачей данных, так и с состоянием систем, участвующих в интеграции. Наиболее продвинутые системы предоставляют средства проактивной диагностики. Этот вид диагностики позволяет выявлять потенциальные проблемы на начальных этапах до того, как проблема проявит себя в полную силу, и своевременно осуществлять комплекс упреждающих воздействий.

2.2 Выбор веб-сервиса обмена сообщениями. Краткая характеристика

В качестве брокера сообщений был выбран RabbitMq.

RabbitMQ – это брокер сообщений с открытым исходным кодом. Он маршрутизирует собщения по всем базовым принципам протокола AMQ.

RabbitMQ передаёт сообщения между поставщиками и подписчиками через очереди. Сообщения могут содержать любую информацию, например, о событии, произошедшем на сайте.

RabbitMQ отлично подходит для интеграции разных компонентов, создания микросервисов, потоковой передачи данных в режиме реального

времени или при передаче работы удалённым работникам. Его используют крупные компании, в числе которых Bloomberg, Reddit, WeWork, NASA и др.

2.3 Определение формата данных

В качестве формата данных будет использоваться CSV (Comma-Separated Values), где каждое поле разделяется запятой. Каждое сообщение содержит информацию о студенте и состоит из четырех основных элементов: группа, имя, предмет, балл.

2.4 Определение регламента обмена данными

Обмен данных будет происходить по специальному событию: а именно – открытии формы выставления оценки студенту. После того, как администратор открыл эту форму, из очереди достается сообщение и данные автоматически заполняются в нужные строки.

3 Реализация и развертывание интеграционного решения

3.1 Настройка веб-сервиса обмена сообщениями

Для настройки очереди достаточно подключить библиотеку (RabbitMQ.Client) и написать функции для отправителя и получателя. Кроме того, нужно установить и запустить клиент rabbitmq.

3.2 Информационная система № 1 – общий вид

Главная страница представлена на рисунке 1.

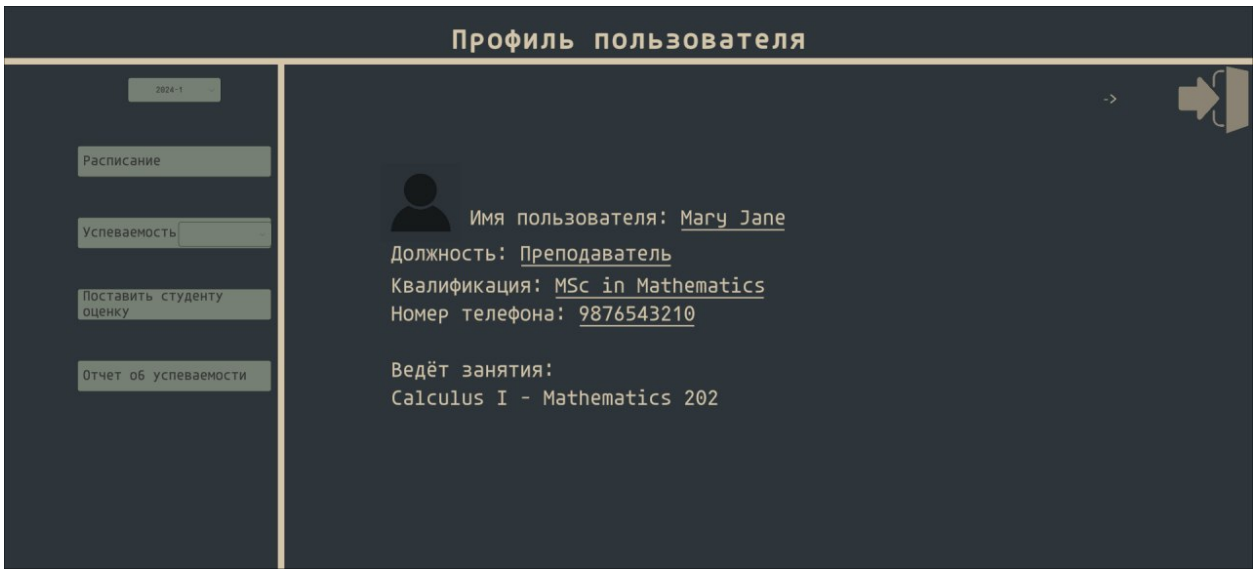

Рисунок 1 - Главная страница

Добавление новой оценки представлено на рисунке 2.

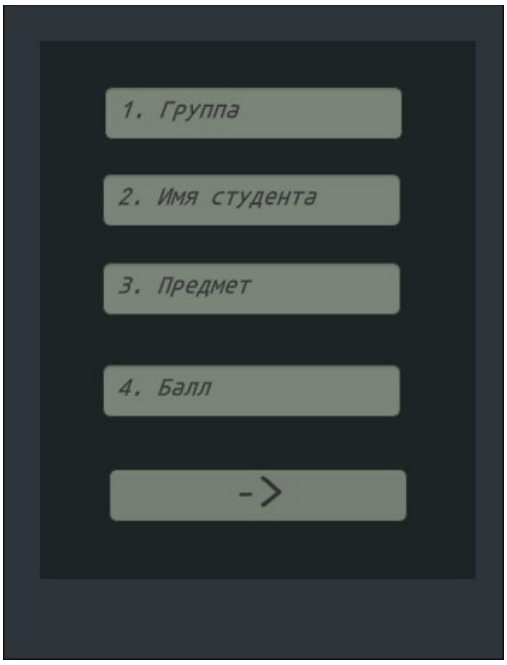

Рисунок 2 - Добавление новой оценки

3.3 Информационная система № 2 – общий вид

Общий вид у ИС №2 отсутствует, т.к. она разрабатывалась в виде консольного приложения.

3.4 Работа интеграционного решения

На рисунке 3 представлен процесс отправки сообщения.

```
stanik@archlinux::/home/stanik/scripts/rust/rabbitmq_sender::MVVM > cargo run
    Finished `dev` profile [unoptimized + debuginfo] target(s) in 0.23s
     Running `target/debug/rabbitmq_sender`
Отправляем CSV: ,"АС-21-1","Иванов","Программирование","100"
CSV-сообщение успешно доставлено в RabbitMQ
Получено сообщение: ,"АС-21-1","Иванов","Программирование","100"
```

Рисунок 3 - Отправка данных

На рисунке 4 представлен процесс приема сообщения.

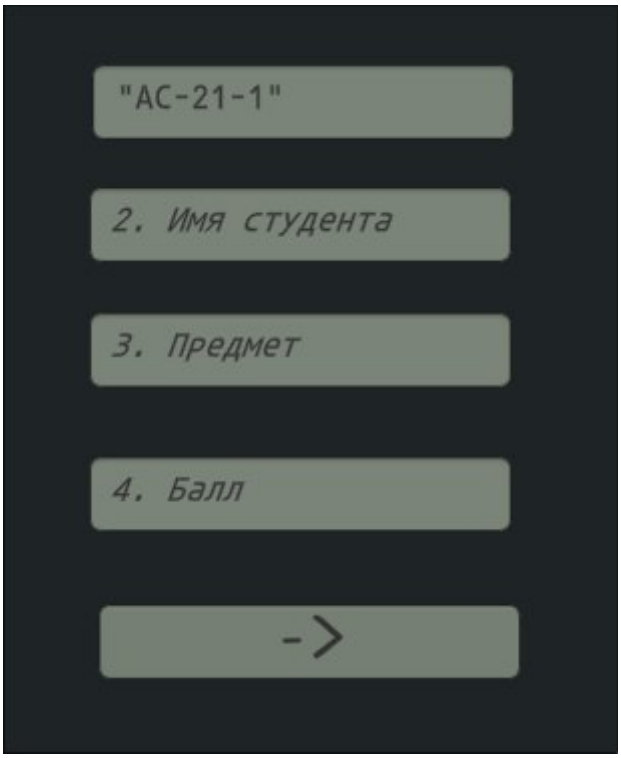

Рисунок 4 - Данные были получены и обновилась группа

Вывод

В данной работе была реализована интеграция между двумя информационными системами для отправки информации о новом клиенте для выставления оценки студенту.

ПРИЛОЖЕНИЕ А

Код отправителя

Код на отправителе (rust)

```
use fe2o3_amqp::types::messaging::Outcome;
use fe2o3_amqp::{Connection, Receiver, Sender, Session};


#[tokio::main]
async fn main() {
    // Устанавливаем соединение с RabbitMQ
    let mut connection = Connection::open(
        "csv-publisher",                    // container id
        "amqp://guest:guest@localhost:5672", // url
    )
    .await
    .unwrap();

    let mut session = Session::begin(&mut connection).await.unwrap();

    // Создаем отправителя
    let mut sender = Sender::attach(
        &mut session,
        "csv-sender-link",
        "csv-queue", // очередь для CSV-сообщений
    )
    .await
    .unwrap();
```

```
    let csv_data = r#","AC-21-1","Иванов","Программирование","100""#;
    println!("Отправляем CSV: {}", csv_data);

    // Отправляем сообщение и ждем подтверждения
    let outcome: Outcome = sender.send(csv_data).await.unwrap();
    match outcome {
        Outcome::Accepted(_outcome) => println!("CSV-сообщение успешно доставлено в RabbitMQ"),
        other => println!("Ошибка доставки: {:?}", other),
    }

    // Закрываем соединения
    sender.close().await.unwrap();
    session.end().await.unwrap();
    connection.close().await.unwrap();
}
```

Код получателя

```
public class RabbitMQReceiver : MonoBehaviour
{
    public Action<CSVMessage> OnMessageGetEvent;
    private IConnection connection;
    private IModel channel;
    private const string queueName = "csv-queue";
    private CSVMessage msg;

    private void Start()
```

```
        {
            // Подключение
            var factory = new ConnectionFactory() { HostName = "localhost", Port = 5672 };
            connection = factory.CreateConnection();
            channel = connection.CreateModel();

            // Очеред
                channel.QueueDeclare(queue: queueName, durable: false, exclusive: false, autoDelete: false, arguments: null);

            // Создание обработчика для получения сообщений
            var consumer = new EventingBasicConsumer(channel);
            consumer.Received += (model, ea) =>
            {
                var body = ea.Body;
                var message = Encoding.UTF8.GetString(body);
                Debug.Log("Received message: " + message);
                ProcessCsvMessage(message);
            };

            // Подписка на очередь
            channel.BasicConsume(queue: queueName, noAck: false, consumer: consumer);
        }
```