

Docente:	Edwin Boza G.	Periodo:	PEL5	Año:	2023
Materia:	Sistemas Distribuidos				
Actividad:	Tarea 02: Comunicación indirecta				

Tarea 02

Diseñar y construir una plataforma tecnológica para la transmisión, almacenamiento y verificación de datos de un censo poblacional

Alumnos:

Isaac Orrala Yungan

Cesar Orrala Yungan

Docente:

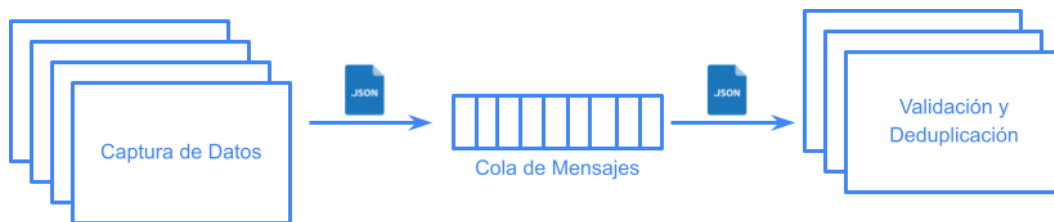
Ing. Edwin Boza G, Phd.

Fecha:

Guayaquil, 05 de noviembre del 2023

Descripción general

Con esta tarea se inicia el desarrollo del proyecto integrador. El objetivo de esta tarea es construir un prototipo funcional de los 2 primeros módulos y la implementación de la comunicación entre ellos a través de una cola de mensajes.



Requerimientos mínimos

1. El módulo de Cola de Mensajes debe ser funcional. Es decir que debe permitir la comunicación efectiva entre los módulos de Captura de Datos y de Validación/Deduplicación.
2. El “Módulo de Captura de Datos” será una aplicación independiente que generará de forma automática los formularios del censo llenos, para lo cual puede llenarlos de forma aleatoria. Cada vez que genere un nuevo formulario, deberá conectarse por medio de la red IP a la cola de mensajes y depositar el formulario codificado en formato JSON.
3. El módulo de Validación/Deduplicación se encargará de obtener los formularios en formato JSON desde la cola de mensajes. Para esta entrega bastará con que obtenga los formularios y realice la validación. *La deduplicación no será requerida para esta entrega debido a que el módulo de almacenamiento aún no está disponible.*

Indicaciones sobre la entrega

- Se debe entregar un reporte indicando el diseño arquitectónico de los módulos implementados, así como una **explicación** de las configuraciones utilizadas para la cola de mensaje.
- Enlace a un video de máximo 10 minutos con la demostración de la ejecución de las aplicaciones desarrolladas.
- Para el desarrollo del proyecto deben utilizar un repositorio en GitHub. Puede ser un repositorio privado compartido con los integrantes del grupo y con el

Docente:	Edwin Boza G.	Periodo:	PEL5	Año:	2023
Materia:	Sistemas Distribuidos				
Actividad:	Tarea 02: Comunicación indirecta				

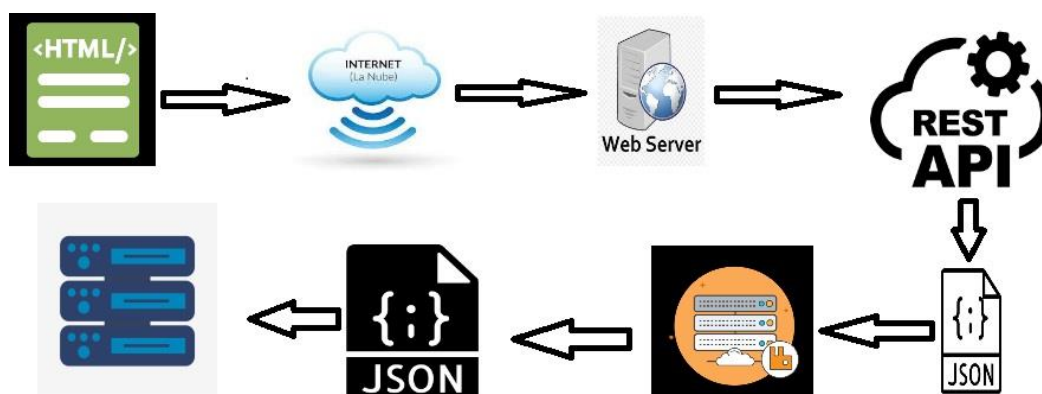
docente (usuario **ebozag**). Recuerden que en el repositorio solo se sube el código fuente, no se suben los archivos compilados o ejecutables.

- El reporte debe incluir los nombres de los integrantes del grupo, así como la URL del repositorio.
- El reporte también debe mostrar capturas del funcionamiento, con una descripción clara de lo que muestra cada imagen.
- El reporte debe ser subido en un archivo PDF.
- No subir en el Blackboard archivos ZIP, ni código fuente, ni en otros formatos. Únicamente el archivo del reporte en formato PDF.
- El formato del reporte es libre, pero debe mostrar profesionalismo y cuidado en el desarrollo, además, las capturas de pantalla deben ser claras e indicar claramente lo que se está mostrando en cada pantalla.
- Se recomienda realizar la tarea con tiempo y subirla al Blackboard lo antes posible para ser evaluada. Las entregas tardías de hasta 2 días tendrán una penalización de 50 puntos. No se aceptan entregas de más de 2 días de retraso.
- Las dudas sobre el trabajo se pueden publicar en el foro.
- También es válido publicar en el foro referencias a los sitios donde encuentren instrucciones que les ayuden a completar la tarea, y de esta manera colaborar con sus compañeros del curso.
- El docente puede solicitar **demostraciones del funcionamiento** durante las sesiones sincrónicas, además de una explicación del código o del procedimiento realizado.

Definición del proyecto

La creación de un sistema censal es el proceso de diseñar, desarrollar e implementar una infraestructura técnica y metodológica que permita recolectar y gestionar datos demográficos y socioeconómicos de una población específica. El objetivo principal del sistema es obtener información precisa y actualizada sobre la composición y características de una población específica.

Arquitectura de la solución propuesta



Diseño y desarrollo del servidor Linux:

Configuración del servidor Linux con los requisitos necesarios, como el sistema operativo, servicios web, base de datos, etc.

Implementación de medidas de seguridad para proteger el servidor y los datos almacenados.

Desarrollo de la aplicación web en Windows:

Utilización de herramientas de desarrollo web, como HTML, CSS y JavaScript, para crear la interfaz de usuario de la aplicación.

Uso de Java para desarrollar la lógica del lado del servidor y la interacción con el servidor Linux

RabbitMQ.

Implementación de la integración con RabbitMQ para el intercambio de mensajes entre la aplicación web y el servidor Linux.

Implementación de RabbitMQ:

Configuración de RabbitMQ como un servicio de mensajería para facilitar la comunicación entre el servidor Linux y la aplicación web en Windows.

Definición de las colas de mensajes y los intercambios necesarios para el procesamiento de datos del censo.

Intercambio de archivos JSON:

Definición de un formato de archivo JSON para el intercambio de datos del censo entre el servidor Linux y la aplicación web.

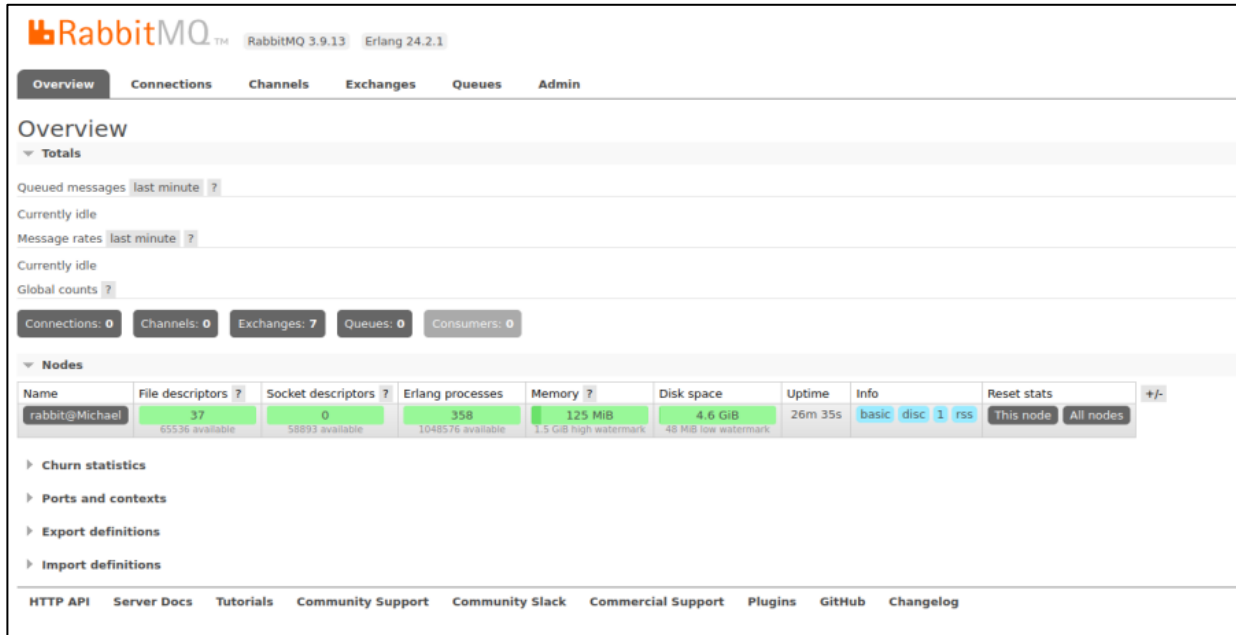
Implementación de mecanismos para enviar y recibir archivos JSON a través de RabbitMQ, asegurando la integridad de los datos durante el proceso.

Registro de logs:

Implementación de un sistema de registro de logs en el servidor Linux y la aplicación web para realizar un seguimiento de las operaciones realizadas y facilitar el proceso de depuración y monitoreo.

Definición de los eventos y mensajes relevantes a registrar, así como el almacenamiento y acceso a los registros generados.

El módulo de Cola de Mensajes



RabbitMQ ™ RabbitMQ 3.9.13 Erlang 24.2.1

Overview | Connections | Channels | Exchanges | Queues | Admin

Overview

Totals

Queued messages last minute ?

Currently idle

Message rates last minute ?

Currently idle

Global counts ?

Connections: 0 Channels: 0 Exchanges: 7 Queues: 0 Consumers: 0

Nodes

Name	File descriptors <small>?</small>	Socket descriptors <small>?</small>	Erlang processes	Memory <small>?</small>	Disk space	Uptime	Info	Reset stats
rabbit@Michael	37 <small>65536 available</small>	0 <small>58893 available</small>	358 <small>1048576 available</small>	125 MiB <small>1.5 GiB high watermark</small>	4.6 GiB <small>48 MiB low watermark</small>	26m 35s	basic disc 1 rss	<small>This node All nodes</small>

Churn statistics

Ports and contexts

Export definitions

Import definitions

[HTTP API](#) [Server Docs](#) [Tutorials](#) [Community Support](#) [Community Slack](#) [Commercial Support](#) [Plugins](#) [GitHub](#) [Changelog](#)

```
JS cola_rabbitjs > ...
11   const auth = {
12     username,
13     password
14   };
15
16   try {
17     const response = await axios.get(`${rabbitmqUrl}/api/queues/${encodeURIComponent(virtualHost)}/${queueName}`, { au
18     const queueData = response.data;
19
20     if (queueData.messages > 0) {
21       console.log(`La cola ${queueName} tiene mensajes pendientes`);
22     } else {
23       console.log(`La cola ${queueName} está vacía`);
24     }
25   } catch (error) {
26     console.error('Error al verificar la cola:', error);
27   }
28 };
29
30 // Llamar a la función para verificar la cola
31 checkQueue();
32
```

```

JS server1.js > ...
39 // Redirige al usuario de vuelta a la página del formulario
40 res.writeHead(200, { 'Content-Type': 'text/html' });
41 res.write('<meta http-equiv="refresh" content="0;URL=\'/\'' />');
42 res.end();
43
44 // Crear archivo con los datos recibidos
45 const fileContent = `{"id": "${timestamp}_${cedula}", "Nombre": "${name}", "Nacionalidad": "${nacionalidad}", "Cedula": "${cedula}"`;
46 fs.appendFile('pasar.json', fileContent, err => {
47   if (err) {
48     console.error('Error al crear el archivo:', err);
49     res.writeHead(500);
50     res.end('Error interno del servidor');
51   } else {
52     console.log('Archivo creado correctamente');
53     res.writeHead(200, { 'Content-Type': 'text/plain' });
54     res.end('Archivo creado correctamente');
55   }
56 });
57
58 });
59 } else {
60   res.writeHead(404);
61   res.end('Página no encontrada');
62 }
63 });
64
65 const port = 3000;
66 server.listen(port, () => {
67   console.log(`Servidor escuchando en http://localhost:${port}`);
68 });
69

```

Servidor de js el cual leer el archivo .json y envía los datos a la cola de mensajes de Rabbit por el canal Monitor “envia_rabbit.js”

```

const amqp = require('amqplib/callback_api');

const options = {
  clientProperties: {
    connection_name: 'producer-service'
  }
};

amqp.connect('amqp://trabajo:trabajo@192.168.100.134', options, (error, connection) => {
  if (error) {
    throw error;
  }

  connection.createChannel((connErr, channel) => {
    if (connErr) {
      throw connErr;
    }

    channel.assertQueue('Monitor', {
      durable: true
    });

    const fs = require('fs');

    // Ruta al archivo que deseas leer
    const rutaArchivo = 'C:/xampp/htdocs/Programacion/pasar.json';

    // Función para leer el archivo
    fs.readFile(rutaArchivo, 'utf8', (error, contenido) => {
      if (error) {
        console.error('Error al leer el archivo:', error);
      }
    });
  });
});

```

Principal

Archivo | C:/Users/Isaac%20Orrala/Downloads/Pasar/Pasar/Programacion/Programacion/index.html

Integrantes del Grupo:

- Isaac Orrala Yungan
- Cesar Orrala Yungan

Formulario de censo poblacional

Nombre completo:	Nacionalidad:
Nombre completo	Nacionalidad
Documento de identidad:	Dirección de residencia:
Documento de identidad	Dirección completa
Numero de teléfono:	Correo electrónico:
Número de telefono	Correo electrónico

Guardar

1. formulario codificado en formato JSON.

```
1 Ecuatoriana","Cedula":"0921406898","Direccion":"Florida Norte","Telefono":"0967843494","Correo":"cesarorrala@uees.edu.ec"}
2 Ecuatoriana","Cedula":"0943942359","Direccion":"San Eduardo","Telefono":"0973822334","Correo":"isaac.orrala@uees.edu.ec"}
```