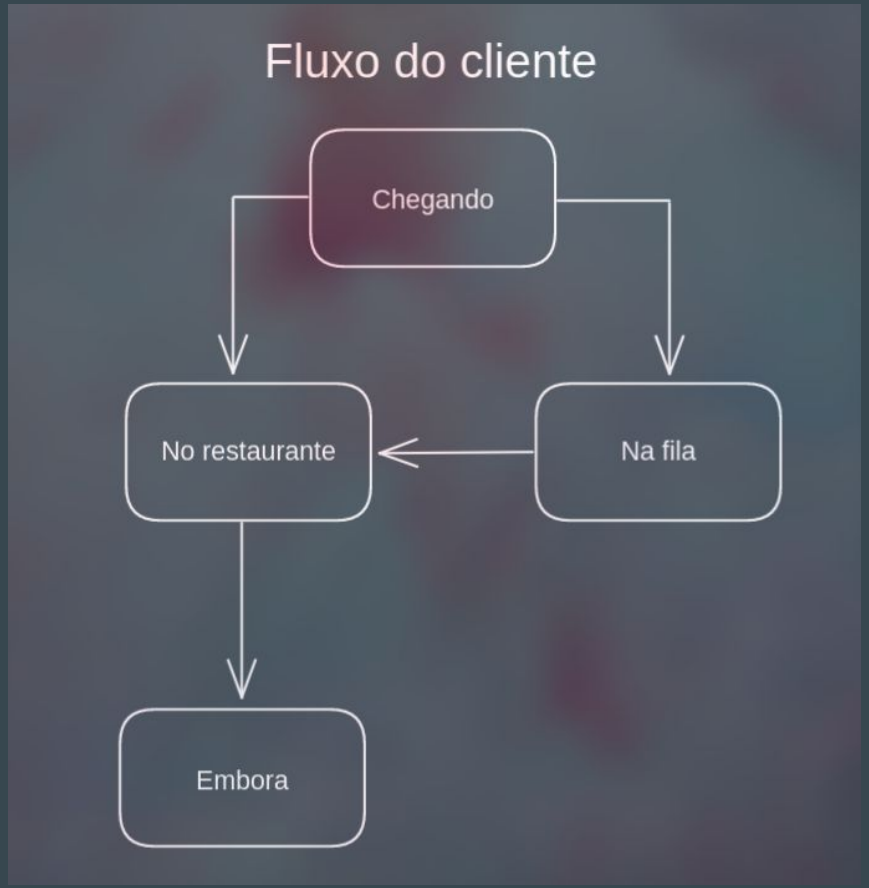
Exercício 8: Sushi Bar

Nilo Bemfica Mineiro Campos Drumond - Pedro Didier Maranhão

Implementação - Rust

Foi usado RabbitMQ (biblioteca amiquip) como middleware para comunicação.

```rust
pub enum CustomerStatus {
    Arriving,
    InQueue,
    Entered,
    Left,
}
```



Configurações

Cliente

```
let mut connection = Connection::insecure_open("amqp://guest:guest@localhost:5672")?; (url)
let channel = connection.open_channel(None)?; (channel_id) Channel
let exchange = Exchange::direct(&channel); Exchange

let queue = channel.queue_declare(format!("sushi/{id}"), QueueDeclareOptions::default())?;
let consumer = queue.consume(ConsumerOptions::default())?; (options) Consumer
```

Servidor

```
let mut connection = Connection::insecure_open("amqp://guest:guest@localhost:5672")?;
let channel = connection.open_channel(None)?; (channel_id) Channel
```

Estrutura de dados (servidor)

```rust
struct Bar {
    table: Vec<i64>,
    queue: Queue<i64>,
    occupied: bool,
}
```

```rust
let bar = Bar { Bar
    queue: queue![],
    table: vec![],
    occupied: false,
};

let bar = Arc::new(Mutex::new(bar));
```

Fluxo do cliente

```rust
let mut status = CustomerStatus::Arriving; CustomerStatus
println!("Cliente {id}: {status}");

exchange.publish(Publish::new(&id.to_be_bytes(), "sushi"))?; (body, routir
for (_, message) in consumer.receiver().iter().enumerate() { ConsumerMessa
    match message {
        amiquip::ConsumerMessage::Delivery(delivery) => { Delivery
            status = CustomerStatus::try_from(delivery.body[0]).unwrap();
            consumer.ack(delivery)?;
            println!("Cliente {id}: {status}");
            if status == CustomerStatus::Left {
                break;
            }
        }
        other => { ConsumerMessage
            println!("Consumer ended: {:?}", other);
            break;
        }
    }
}
```

Fluxo do servidor

```rust
let queue = channel.queue_declare("sushi", QueueDeclareOptions::default())?;
let consumer = queue.consume(ConsumerOptions::default())?; (options) Consumer

for (_, message) in consumer.receiver().iter().enumerate() { ConsumerMessage
    match message {
        amiquip::ConsumerMessage::Delivery(delivery) => { Delivery
            let data = parse_data(&delivery.body[..8]); (raw) [u8; 8]
            consumer.ack(delivery)?;
            let client_id = i64::from_be_bytes(data); i64
            receive_customer(client_id)
        }
        other => { ConsumerMessage
            println!("Consumer ended: {:?}", other);
            break;
        }
    }
}
```

Novo cliente (servidor)

```
let receive_customer = |id: i64| { |i64| -> ()
    let mut bar = bar.lock().unwrap(); MutexGuard<Bar>

    if !bar.occupied {
        bar.table.push(id);
        update_status(&exchange, id, CustomerStatus::Entered);
        if bar.table.len() == 5 {
            bar.occupied = true
        }
    } else {
        bar.queue.add(id).unwrap(); (val)
        update_status(&exchange, id, CustomerStatus::InQueue);
    }
};
```

Cliente saindo (servidor)

```
loop {
    let mut rng = rand::thread_rng(); ThreadRng
    let should_leave: bool = rng.gen();
    if should_leave {
        let mut bar = bar.lock().unwrap(); MutexGuard<Bar>

        if let Some(id) = bar.table.pop() { i64
            update_status(&exchange, id, CustomerStatus::Left); (status)
            if bar.table.is_empty() {
                bar.occupied = false;
            }

            if !bar.occupied {
                while let Ok(id) = bar.queue.remove() { i64
                    bar.table.push(id);
                    update_status(&exchange, id, CustomerStatus::Entered);
                    if bar.table.len() == 5 {
                        bar.occupied = true;
                        break;
                    }
                }
            }
        }
    }
    thread::sleep(Duration::from_secs(2)); (dur)
}
```

Exemplo

```
n
Cliente 1: Chegando
Cliente 1: Dentro
n
Cliente 2: Chegando
Cliente 2: Dentro
n
Cliente 3: Chegando
Cliente 3: Dentro
n
Cliente 4: Chegando
Cliente 4: Dentro
n
Cliente 5: Chegando
Cliente 5: Dentro
Cliente 5: Saiu
```

```
n
Cliente 6: Chegando
Cliente 6: Na fila
n
Cliente 7: Chegando
Cliente 7: Na fila
Cliente 4: Saiu
Cliente 3: Saiu
Cliente 2: Saiu
Cliente 1: Saiu
Cliente 7: Dentro
Cliente 6: Dentro
Cliente 7: Saiu
Cliente 6: Saiu
```