POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

EICTA – Second Project Delivery – Neo4j and BPMN

Author(s): **Gabriella Bertolino**
Sofia Bulletti
Francesco Friolo
Domenico Pittari

Group Number: **05**

Academic Year: 2025-2026

Contents

1 | Neo4j

1.1. Chosen Nodes Representation

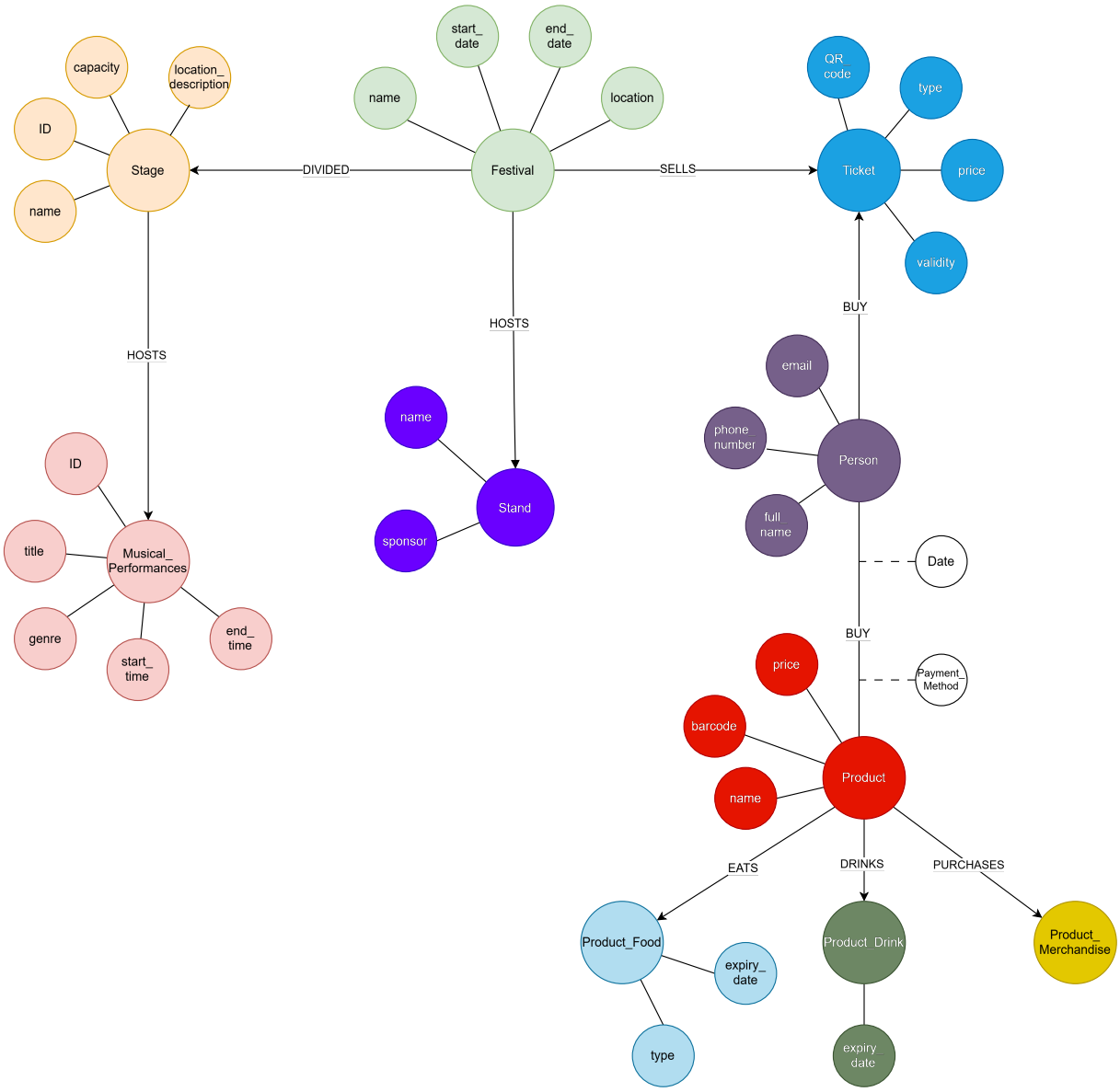


Figure 1.1: Neo4j nodes, relationships and attributes representation

1.2. Nodes, Relationships and Properties Description

The node types and their properties are described below:

- **Festival**: it represents the overall festival event.
  Properties: *name, start_ date, end_ date, location.*
- **Stage**: it defines physical areas or locations within the festival.
  Properties: *ID, name, capacity, location_ description.*
- **Musical_Perfomances**: it represents performances taking place on stages.
  Properties: *ID, title, genre, start_ time, end_ time.*
- **Stand**: it represents commercial or food stands available at the festival.
  Properties: *name, sponsor.*
- **Ticket**: it represents entry tickets to the festival.
  Properties: *QR_ code, type, price, validity.*
- **Person**: it represents customers.
  Properties: *email, phone_ number, full_ name.*
- **Product**: Generic product entity, specialized into three subtypes.
  Properties: *price, barcade, name.*
  - **Food**: Properties: *expiry_ date, type.*
  - **Drink**: Properties: *expiry_ date.*
  - **Merchandise**

The relationship types are described below:

- Festival DIVIDED INTO → Stage (Each festival is divided across several stages).
- Festival HOSTS → Stand (Stands are part of the festival setup).
- Stage HOSTS → Musical_Performances (Each stage hosts one or more performances).
- Festival SELLS → Ticket (Tickets are sold for entry to the festival).
- Person BUYS → Ticket (People purchase tickets to attend)

- Person BUYS → Product (Attendees purchase food, drinks, or merchandise.) Properties: *payment_method, date*
- Product EATS / DRINKS / PURCHASES → Product_Food, Product_Drink, Product_Merchandise (Subtype relationships identifying specific product categories.)

1.3. Queries

1.3.1. Festival–Stand Relationship (CREATE)

Creates a HOSTS relationship between the festival Austin City Limits and the stand Barton Springs Bar, then returns the festival, relationship, and stand.

```
MATCH (f: Festival), (s: Stand)
WHERE f.name= "Austin City Limits" AND s.name= "Barton Springs Bar"
CREATE (s) <- [r: HOSTS] - (f)
RETURN f, r, s;
```

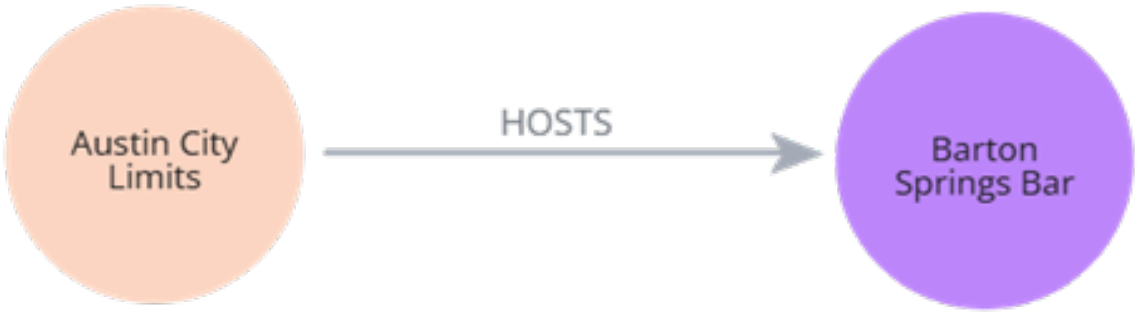


Figure 1.2: Festival–Stand Relationship

1.3.2. New Person Node (CREATE)

Creates a Person node for Charlotte Gonzalez with her email and phone number, then returns the new person node.

```
CREATE (p: Person {email: "charlotte.gonzalez@email.com", full_name: "Charlotte Gonzalez", phone_number: "+1-555-0109"})
RETURN p
```

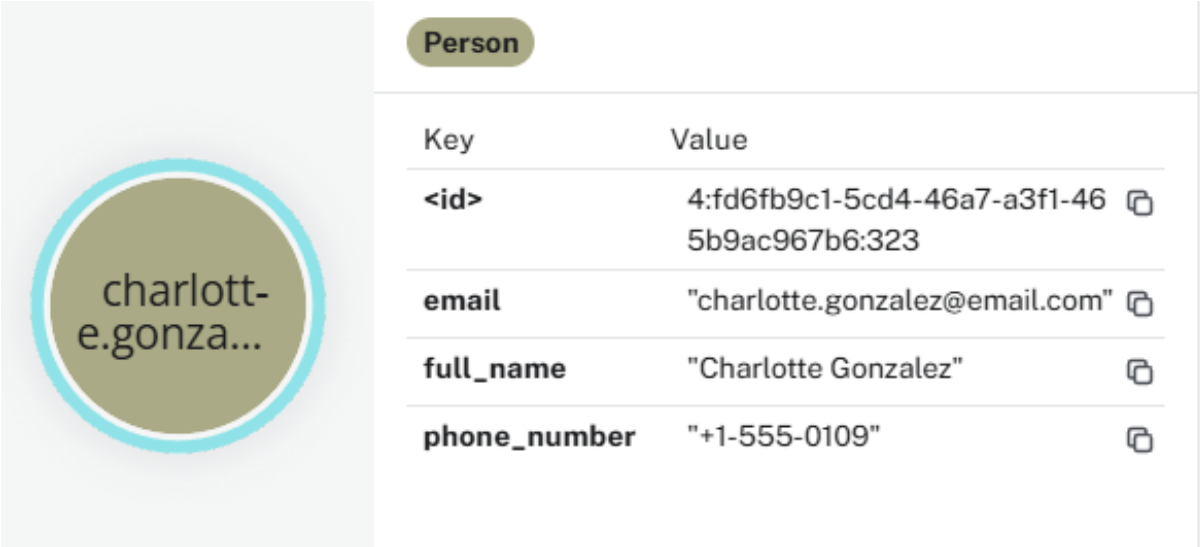


Figure 1.3: New Person Node

1.3.3. Ticket Information Update (UPDATE)

Finds the Ticket with QR code 'QR031BON2024', updates its price to 370 and its type to Regular, then returns the updated ticket.

```
MATCH (t: Ticket)
WHERE t.qr_code= "QR031BON2024"
SET t.price = 370, t.type="Regular"
```

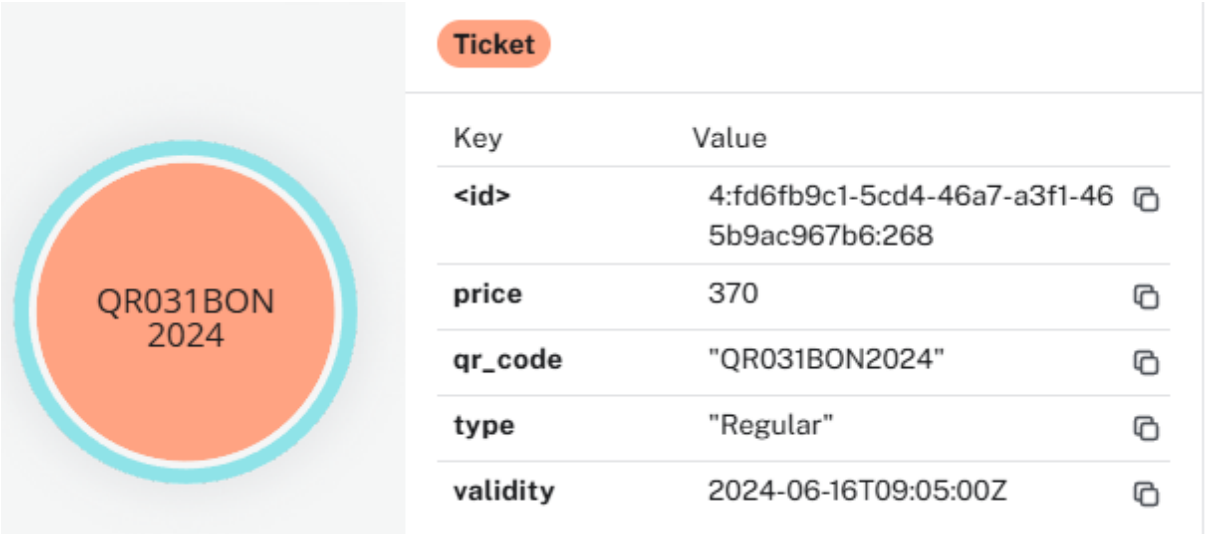


Figure 1.4: Ticket Information Update

1.3.4. Person Contact Update (UPDATE)

Finds the Person with email john.smith@email.com , updates their phone number to +1-555-0101, and returns the person node.

```
MATCH (p: Person {email: "john.smith@email.com"})
SET p.phone_number = "+1-555-0101"
RETURN p;
```

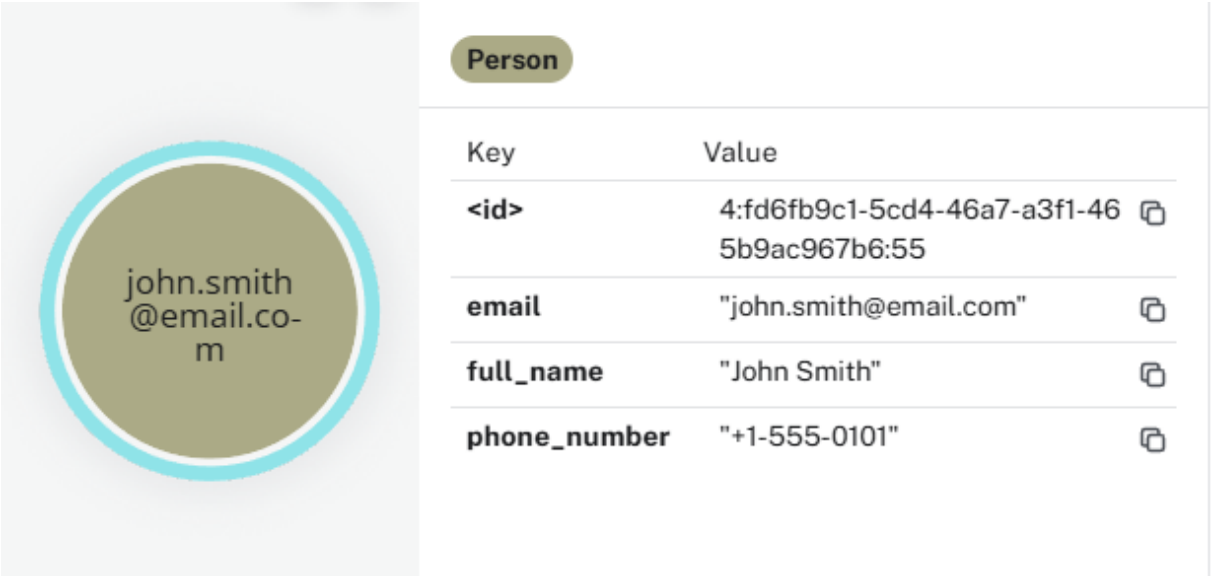


Figure 1.5: Person Contact Update

1.3.5. Delete Stage and its relationships (DELETE)

Deletes the Stage node named 'Main Stage East' and removes all its connected relationships using DETACH DELETE.

```
MATCH (s: Stage {name: "Main Stage East"})
DETACH DELETE s;
```

This query results in the removal of the 'Main Stage East' node along with its three connected relationships.

1.3.6. Delete 'Vends' Relationship (DELETE)

Deletes the VENDS relationship between the stand Carnival Square Food Court and the food product with barcode 1026.

```
MATCH (n: Stand {name: "Carnival Square Food Court"}) - [v: VENDS] -> (f: Product: Food {barcode: 1026})
DELETE v
```

The result of this query is the deletion of the 'vends' relationship between the nodes 'Carnival Square Food' and 'barcode: 1026'.

1.3.7. Festivals in Austin and Hosted Stands (2 NODES IN THE MATCH STATEMENT)

Returns all festivals in Austin, Texas with the stands they host and each stand's sponsor, ordered by festival and stand name.

```
MATCH (f: Festival)-[: HOSTS]->(st: Stand)
WHERE f.location = 'Austin, Texas'
RETURN f.name AS festival,
st.name AS stand,
st.sponsor AS sponsor
ORDER BY festival, stand;
```

festival	stand	sponsor
Austin City Limits	Barton Springs Bar	Tito Vodka

Table 1.1: Festivals in Austin and Hosted Stands

1.3.8. VIP Ticket Buyers (2 NODES IN THE MATCH STATEMENT)

Lists all people who bought VIP tickets, showing their name, email, ticket QR code, and price, ordered by person name.

```
MATCH (p: Person)-[: BUY]->(t: Ticket)
WHERE t.type = 'VIP'
RETURN p.full_name AS person,
p.email AS email,
t.qr_code AS qr_code,
t.price AS price
ORDER BY person;
```

person	email	qr_code	price
Andrew Martin	andrew.martin@email.com	QR020LOL2024	700
Charlotte Gonzalez	charlotte.gonzalez@email.com	QR013TML2024	850
Michael Brown	michael.brown@email.com	QR003GLB2024	750
Nicholas Walker	nicholas.walker@email.com	QR028EDC2024	900
Robert Thomas	robert.thomas@email.com	QR008COA2024	999
Ryan Robinson	ryan.robinson@email.com	QR024READ2024	650
Scarlett Wright	scarlett.wright@email.com	QR033BON2024	780

Table 1.2: VIP Ticket Buyers

1.3.9. Festivals After 01.06.24 and their stands (2 NODES IN THE MATCH STATEMENT WITHOUT A WITH STATEMENT)

Returns all festivals starting on or after 2024-06-01 with the number of hosted stands, ordered by the count of stands descending.

```
MATCH (f: Festival)-[: HOSTS]->(st: Stand)
WHERE date(f.start_date) >= date('2024-06-01')
RETURN f.name AS festival,
count(st) AS num_stands
ORDER BY num_stands DESC;
```

festival	num_stands
Glastonbury Music Festival	5
Bonnaroo Music Festival	4
Burning Man	4
Lollapalooza	4
Reading and Leeds Festival	4
Tomorrowland	4
Austin City Limits	1

Table 1.3: Festivals After 01.06.24 and their stands

1.3.10. Average Ticket Price (2 NODES IN THE MATCH STATEMENT WITHOUT A WITH STATEMENT)

Calculates the average ticket price for tickets valid on or after 2024-01-01 and lists festivals ordered by highest average price.

```
MATCH (f: Festival) - [: SELLS] -> (t: Ticket)
WHERE date(t.validity) >= date('2024-01-01')
RETURN f.name AS festival,
avg(t.price) AS avg_ticket_price
ORDER BY avg_ticket_price DESC;
```

festival	avg_ticket_price
Burning Man	755.0
Coachella Valley Music Festival	705.8
Electric Daisy Carnival	592.0
Glastonbury Music Festival	576.0
Bonnaroo Music Festival	524.0
Tomorrowland	442.5
Lollapalooza	370.0
Reading and Leeds Festival	337.5

Table 1.4: Average Ticket Price

1.3.11. Festival Revenues (2 NODES IN THE MATCH STATEMENT AND 1 WITH STATEMENT)

Sums all ticket revenues for each festival, keeps only those with total revenue above 1400, and orders them by revenue descending.

```
MATCH (f: Festival) - [: SELLS] -> (t: Ticket)
WITH f, sum(t.price) AS revenue
WHERE revenue > 1400
RETURN f.name AS festival, revenue
ORDER BY revenue DESC;
```

festival	revenue
Coachella Valley Music Festival	3529
Electric Daisy Carnival	2960
Glastonbury Music Festival	2880
Bonnaroo Music Festival	2620
Burning Man	2265
Tomorrowland	1770
Lollapalooza	1480

Table 1.5: Festival revenues

1.3.12. Tickets Sold per Festival (2 NODES IN THE MATCH STATEMENT AND 1 WITH STATEMENT)

Counts how many tickets each festival sold, filters for festivals with more than 3 tickets, and orders them by total tickets sold.

```
MATCH (f: Festival) - [: SELLS] -> (t: Ticket)
WITH f, count(t) AS total_tickets
WHERE total_tickets > 3
RETURN f.name AS festival, total_tickets
ORDER BY total_tickets DESC;
```

festival	total_tickets
Bonnaroo Music Festival	5
Coachella Valley Music Festival	5
Electric Daisy Carnival	5
Glastonbury Music Festival	5
Lollapalooza	4
Reading and Leeds Festival	4
Tomorrowland	4

Table 1.6: Tickets Sold per Festival

1.3.13. Revenue and Buyer Count per Festival (3 NODES IN THE MATCH STATEMENT AND MULTIPLE WITH STATEMENTS)

For valid tickets after 2024-01-01, calculates each festival's total revenue and number of unique buyers, listing only festivals with positive revenue ordered by total revenue.

```
MATCH (f: Festival) - [: SELLS] -> (t: Ticket) <- [: BUY] - (p: Person)
WITH f, t, p
WHERE date(t.validity) >= date('2024-01-01')
WITH f, sum(t.price) AS total_revenue,
COUNT(DISTINCT p) AS num_buyers
WHERE total_revenue > 0
RETURN f.name AS festival, total_revenue, num_buyers
ORDER BY total_revenue DESC;
```

festival	total_revenue	num_buyers
Coachella Valley Music Festival	3529	5
Electric Daisy Carnival	2960	5
Glastonbury Music Festival	2880	5
Bonnaroo Music Festival	2620	5
Burning Man	2265	3
Tomorrowland	1770	4
Lollapalooza	1480	4
Reading and Leeds Festival	1350	4

Table 1.7: Revenue and Buyer Count per Festival

1.3.14. Total Spending per Person (3 NODES IN THE MATCH STATEMENT AND MULTIPLE WITH STATEMENTS)

Finds people who bought tickets costing more than 500, calculates their total spending per festival, and returns the person name, festival, and amount spent (limited to 5 results).

```
MATCH (p: Person) - [: BUY] -> (t: Ticket)
WHERE t.price > 500
WITH p, t, t.price AS ticket_price
MATCH (t) <- [: SELLS] - (f: Festival)
WITH p, f, sum(ticket_price) AS total
RETURN p.full_name AS Person, f.name AS Festival, total AS Amount_Spent
LIMIT 5
```

Person	Festival	Amount_Spent
Michael Brown	Glastonbury Music Festival	750
Sophia Davis	Glastonbury Music Festival	1200
Robert Thomas	Coachella Valley Music Festival	999
Isabella Garcia	Coachella Valley Music Festival	1500
Charlotte Gonzalez	Tomorrowland	850

Table 1.8: Total Spending per Person

1.3.15. Strongly Connected People (VARIABLE-LENGTH PATH IN THE MATCH STATEMENT)

Compares every pair of people to find multiple connection paths (length 2–4), returns pairs with at least 3 paths, and shows the shortest path length for the top 10 pairs.

```
MATCH path = (p1: Person) - [*2..4] - (p2: Person)
WHERE p1 <> p2
AND p1.email < p2.email
WITH p1, p2, count(path) AS connections, min(length(path)) AS shortest
WHERE connections >= 3
RETURN p1.full_name AS Person1,
p2.full_name AS Person2,
shortest AS ShortestPath
LIMIT 10
```

Person1	Person2	ShortestPath
Abigail Thompson	Andrew Martin	4
Alexander Allen	Avery Hall	4
Abigail Thompson	Christopher White	4
Andrew Martin	Christopher White	4
Ava Martinez	Christopher White	4
Christopher White	Daniel Wilson	4
Andrew Martin	David Rodriguez	4
Christopher White	David Rodriguez	4
Christopher White	Elizabeth Clark	4
Abigail Thompson	Emily Martinez	4

Table 1.9: Strongly Connected People

1.3.16. Shortest path between two people (SHORTEST PATH)

Finds and returns the shortest path between John Smith and Michael Brown based on their email addresses.

```
MATCH path = SHORTEST( p1: Person {email: 'john.smith@email.com'}) - [ * ] - (p2:
Person {email: 'michael.brown@email.com'})
RETURN path;
```

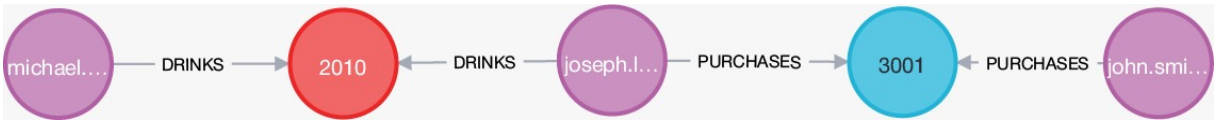


Figure 1.6: Shortest path between two people

2 | BPMN

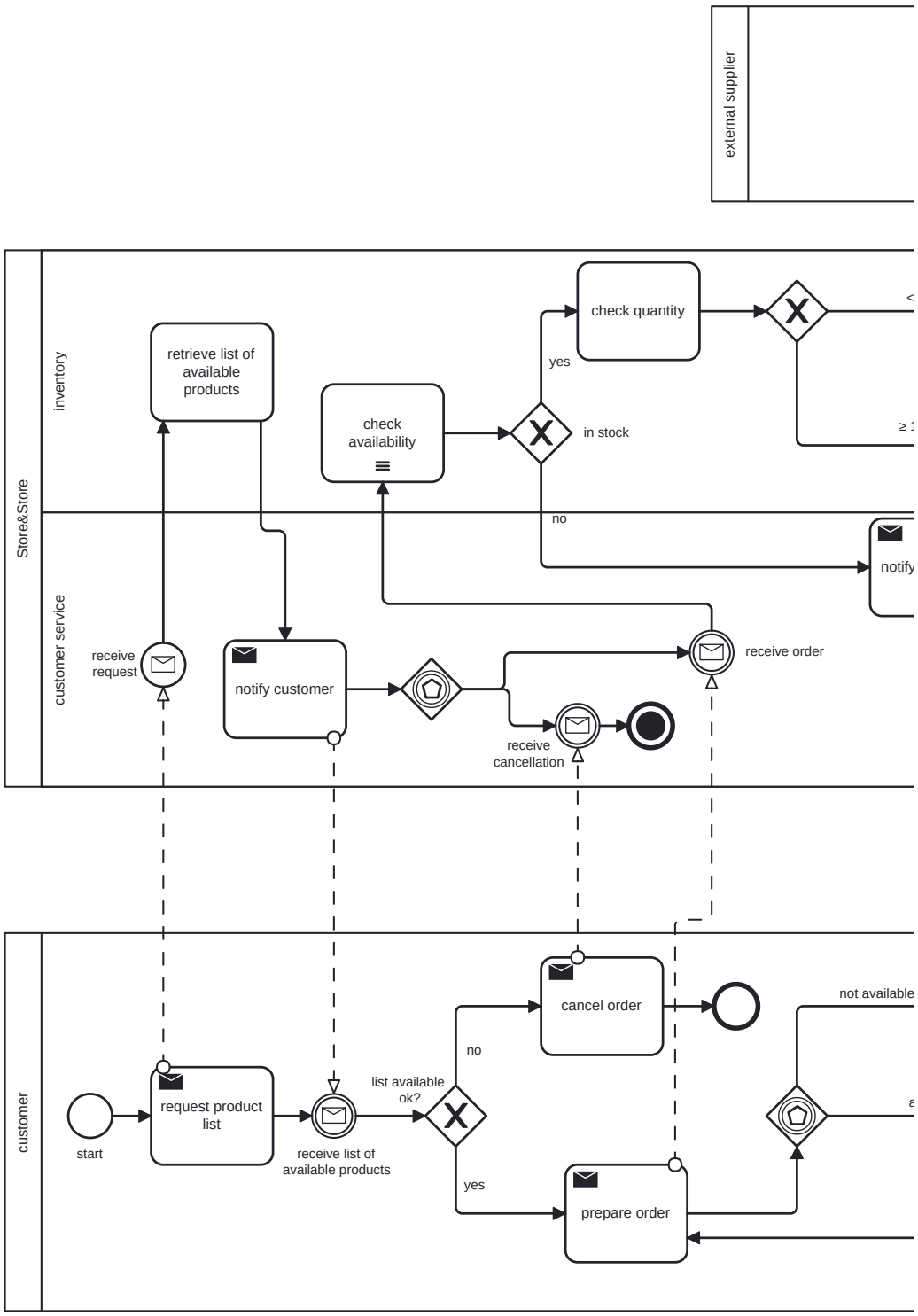
2.1. Provided Process

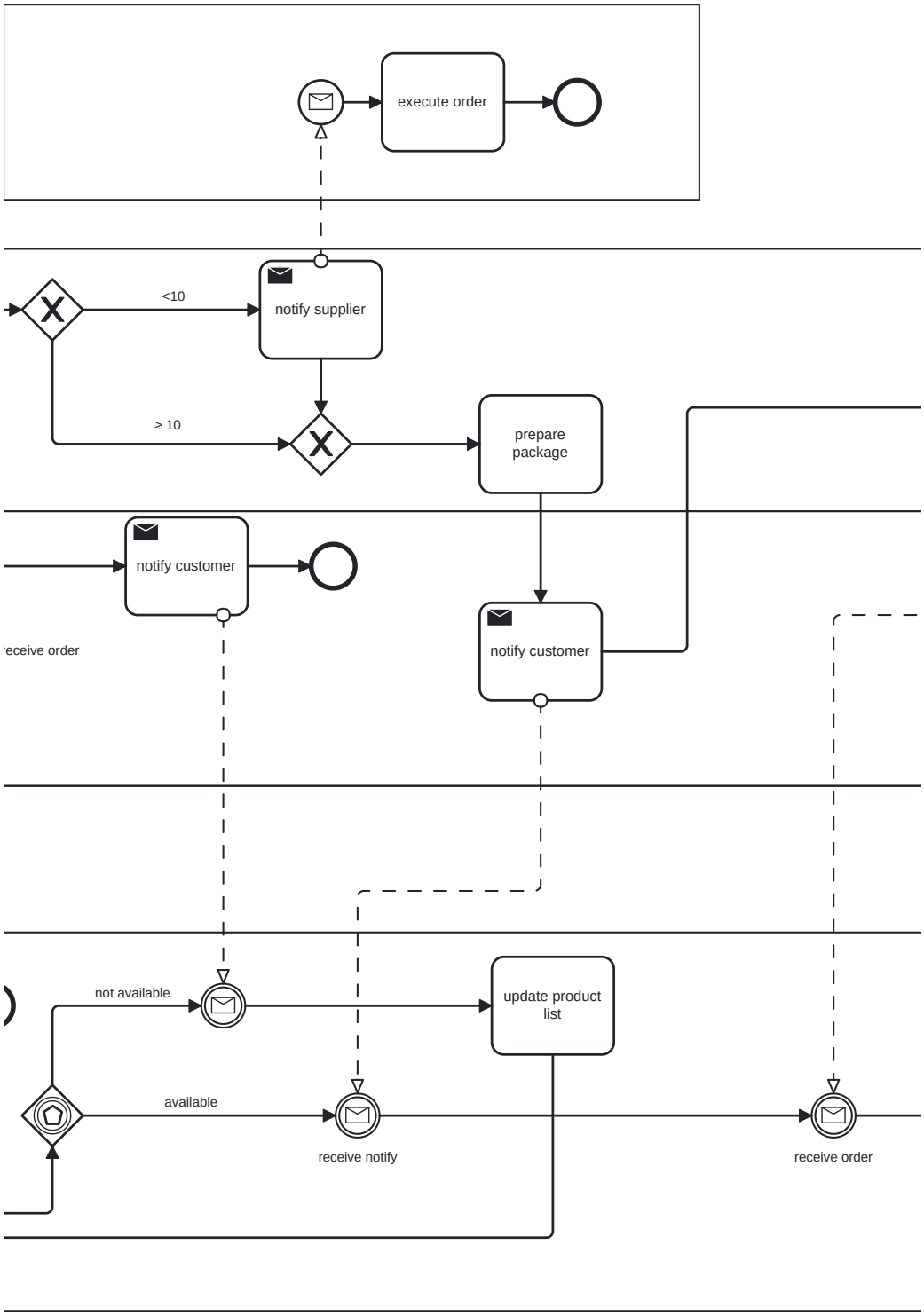
The Store&Store Company is interested in developing an e-commerce system for their electronic stores, and asked you to provide them with a BPMN diagram depicting the pipeline of interactions with the customer, and the preparation and shipping of an order.

The process starts with the customer asking for the available products. The customer service receives the request, and forwards it to the inventory management, which retrieves the list of available products. After receiving the list, the customer prepares its order and sends it to the customer service. The preparation of the order is handled by the inventory management, which checks the availability of each product and, when available, handles its packaging. If any product is not in stock, the customer service sends a notification to the customer, which is required to update the shopping list.

Additionally, to allow better management of the warehouse stocks, when a product is running out (i.e., less than 10 items are available), the inventory management places an order to an external supplier before continuing with the order preparation. After all products of the shopping list have been packaged, the customer service notifies the customer that the order is ready for shipping, and then the order is physically sent by the logistics department. Since it is possible that an order may contain defective products, customers are allowed to return items to the customer service. If no item is returned within a week, the order is recorded and the process terminates. Otherwise, the customer service writes a report for the returned items and the process terminates.

2.1.1. BPMN Diagram




execute order
<10
notify supplier
≥ 10
prepare package
notify customer
eceive order
notify customer
not available
update product list
available
receive notify
receive order

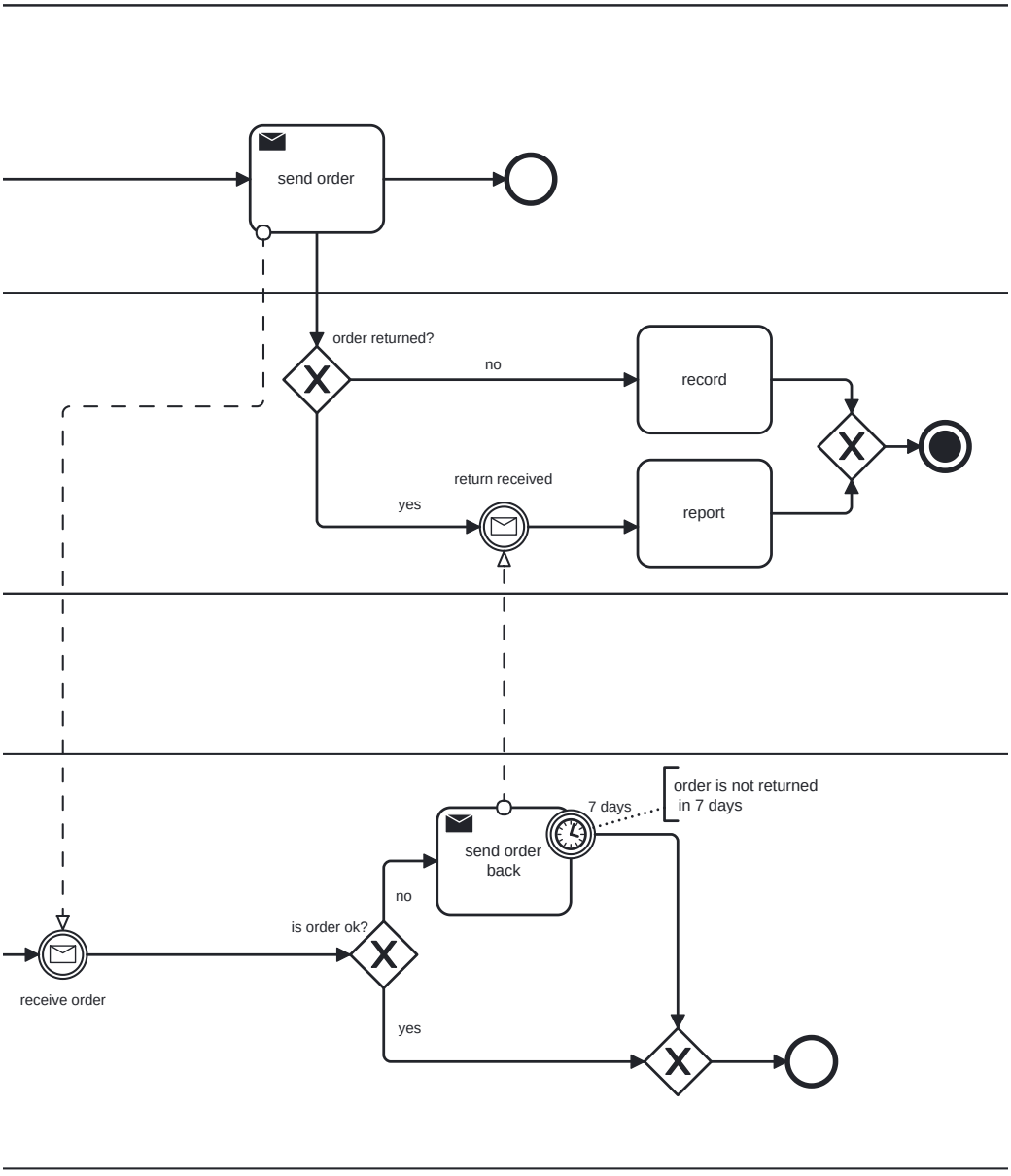


Figure 2.1: BPMN Diagram 1

2.1.2. Assumptions and Reasonings

The following assumptions have been made to support the interpretation and modeling of the BPMN diagrams. They clarify specific behaviors, process flows, and decision points that were not explicitly detailed in the original text.

- We assumed that the order verification process follows a serial sequence, meaning that the individual items within the order are checked one at a time by a staff member rather than simultaneously.
- If the customer receives a list of available products and is not satisfied with the options, they can choose to cancel the order. In this case, a cancellation notification is automatically sent to the Customer Service.
- Once the order has been forwarded to the external supplier, it is assumed that the supplier proceeds with the execution and fulfillment of the order.

2.2. Our designed process

The Euphoric Events Office aims to design a digital system to manage the online sale and validation of event tickets.

The process begins with the customer requesting information about the availability of a festival. Once the request is sent, the Events Office receives it and checks the festival availability. If the request is accepted, the customer receives a confirmation. Otherwise, the customer can submit an alternative request.

After receiving the confirmation, the customer proceeds to choose the ticket type. The Events Office verifies the availability and, if successful, the customer receives a confirmation. Otherwise, the customer can choose another ticket type. The customer then provides personal information and the system proceeds to store the account in the database. Afterwards, the Events Office sends the ticket price to the customer.

The customer is required to complete the payment by entering their card details and authorizing the transaction. If the payment is not successful, the customer must re-enter the card information. However, if the payment is not completed within five minutes, the process terminates with an error.

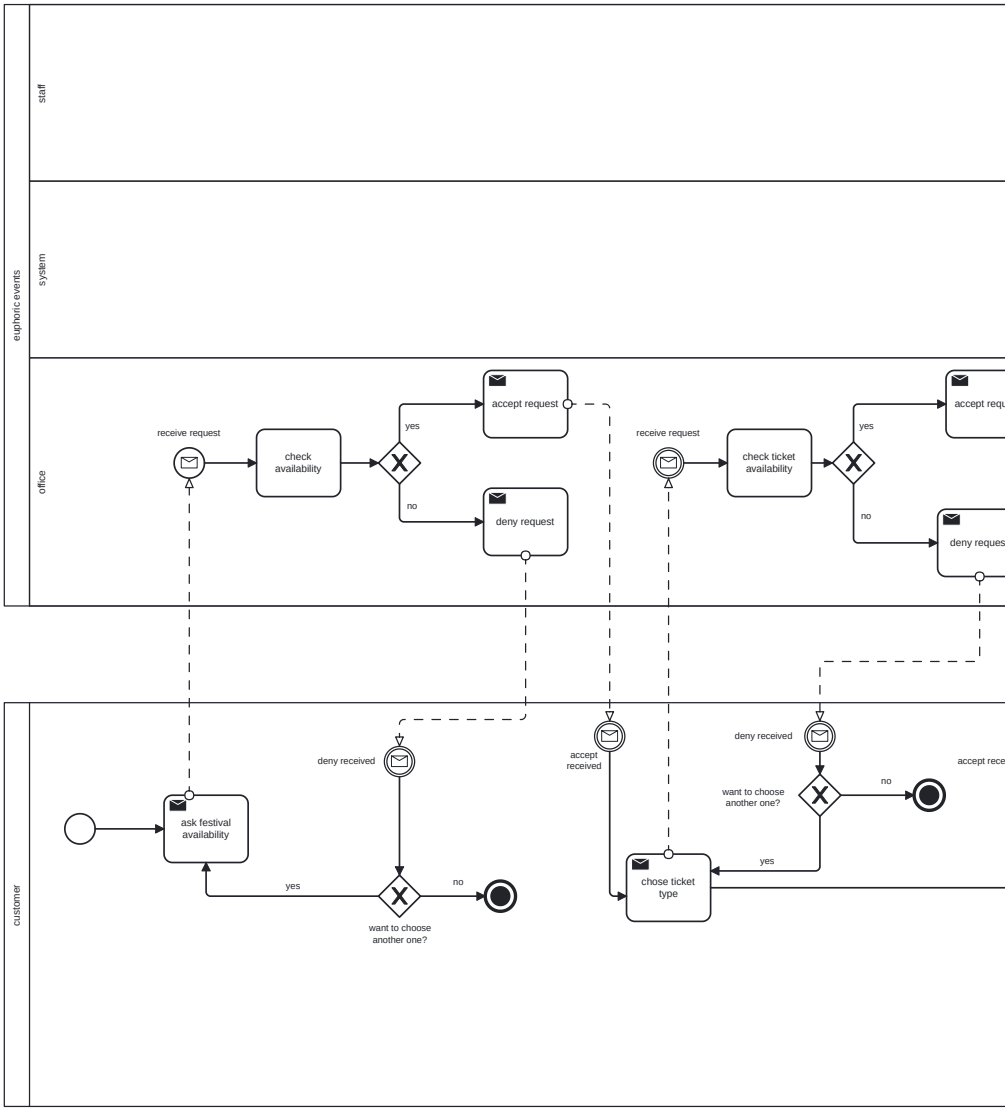
Once the payment is received, the Office sends the information to the database system and the ticket to the customer.

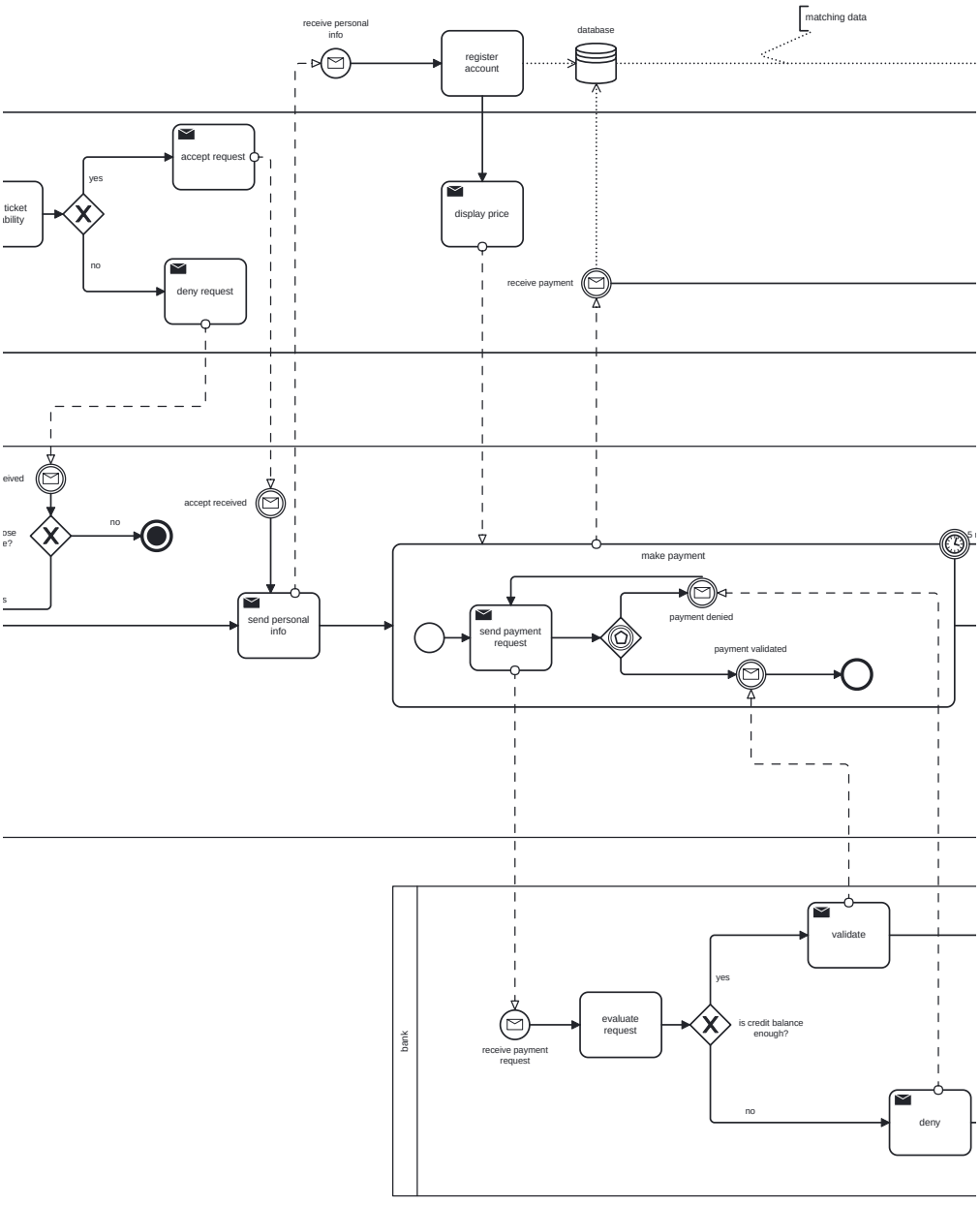
On the day of the event, the ticket must be validated. The staff of the Euphoric Event

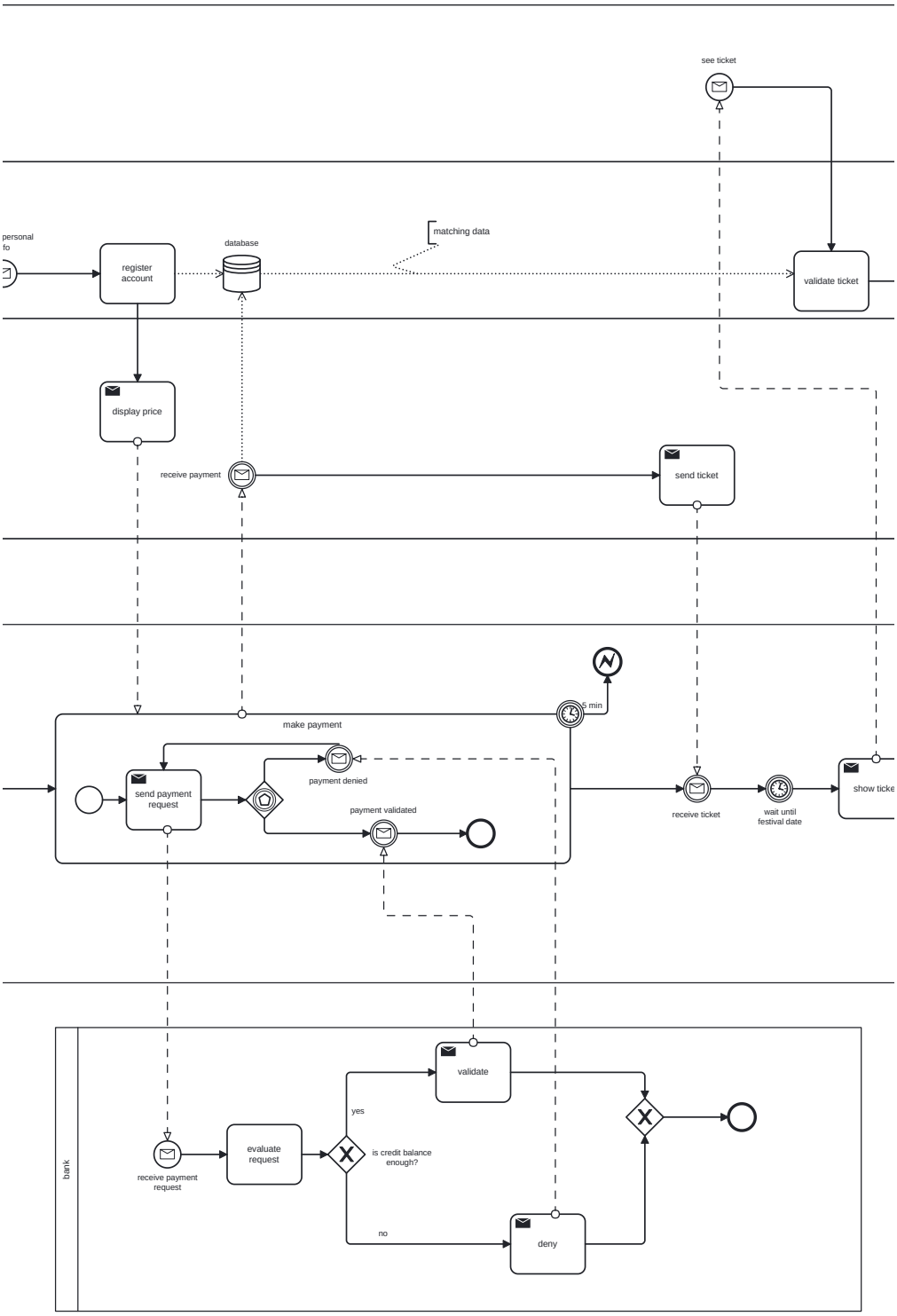
checks, with the help of the information saved in the database, the validity of the ticket. If it is valid, the approval is granted, and the customer is allowed to enter the venue. If the ticket is not valid, the request is rejected and the customer cannot enter.

The process concludes when the customer either successfully presents a valid ticket and gains access to the event, or when the presented ticket is not valid and the customer is required to leave.

2.2.1. BPMN Diagram




receive personal info
register account
database
matching data
accept request
yes
display price
no
deny request
receive payment
accept received
no
send personal info
make payment
send payment request
payment denied
payment validated
bank
receive payment request
evaluate request
validate
yes
is credit balance enough?
no
deny


see ticket
personal
fo
register
account
database
matching data
validate ticket
display price
receive payment
send ticket
5 min
make payment
payment denied
send payment
request
payment validated
receive ticket
wait until
festival date
show ticke
bank
validate
yes
receive payment
request
evaluate
request
is credit balance
enough?
no
deny

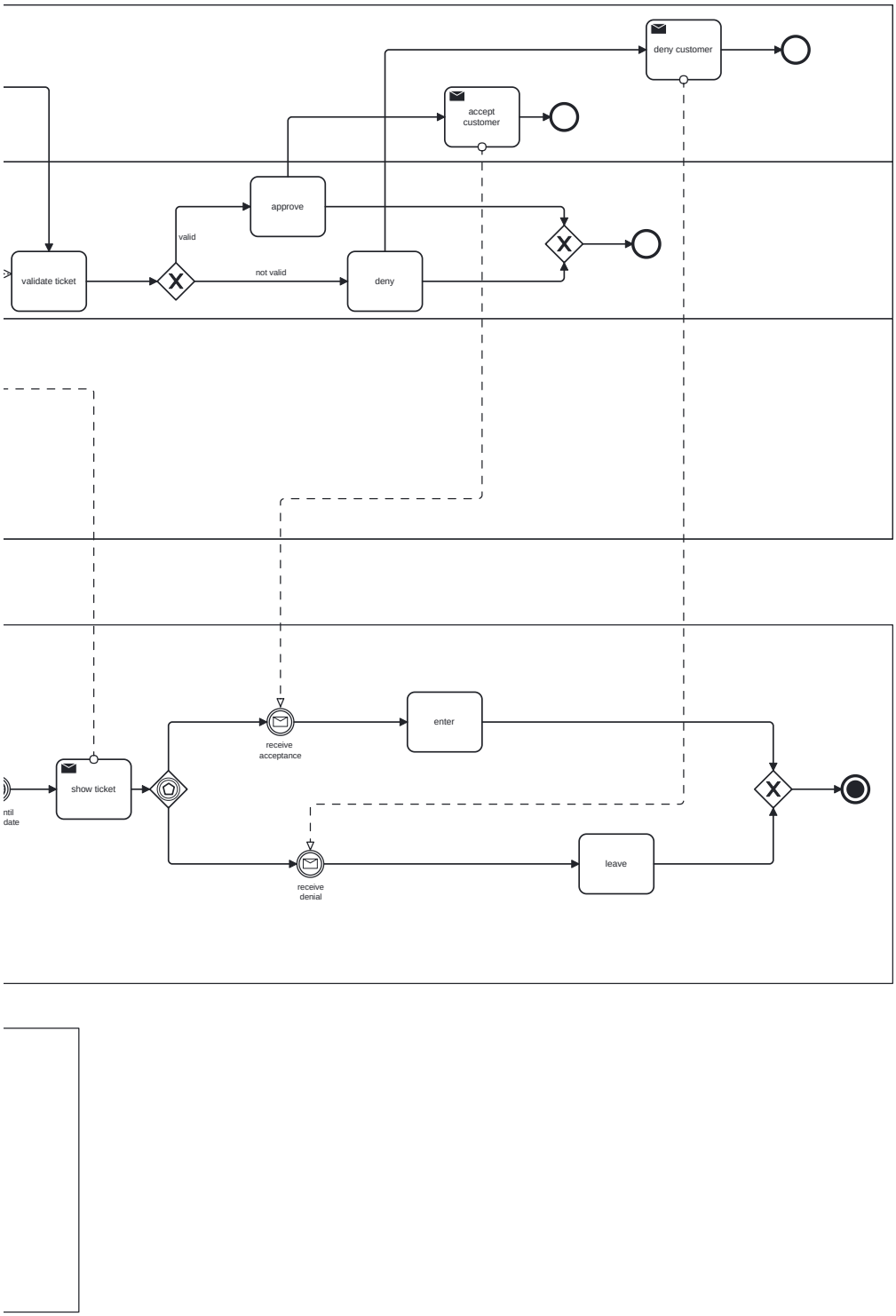


Figure 2.2: BPMN Diagram 2

2.2.2. Assumptions and Reasonings

These assumptions aim to clarify the logical flow and interactions within the BPMN diagrams, ensuring consistency and completeness in the process.

- If the payment is not completed within 5 minutes, an error message is displayed to the customer, indicating a failed or expired transaction.
- In order to validate the ticket, the staff retrieves and cross-checks ticket information directly from the database.
- The subprocess represents the payment request that is sent to the bank. If the payment is successfully processed, the confirmation is forwarded to the office and recorded in the system database.
- If a specific festival or ticket is not available, the customer is allowed to select an alternative option.