Bookshop Atlas

2026-11-19 · `examples/bookshop-atlas.json`

37 views · 312 cards · 306 lines

Contents

Bookshop 3
The whole bookshop business. Double-click a card to go in: each opens a diagram of its part, and those open further.

- Bookshop: system context 5
  - Shopping 6
  - Bookshop: containers 8
    - Shop API: components 10
      - Placing an order 12
  - Buying a book 14
  - Find a book 15
  - Check out 16
  - Email the receipt 17
  - Queue the order 18
  - Bookshop components 19
  - Sales domain 20
    - Payments only 22
    - An order's life 23
      - Order 1042, just placed 25
    - Bookshop packages 26
      - Bookshop conventions profile 28
  - Inside OrderService 30
  - What travels with an order 31

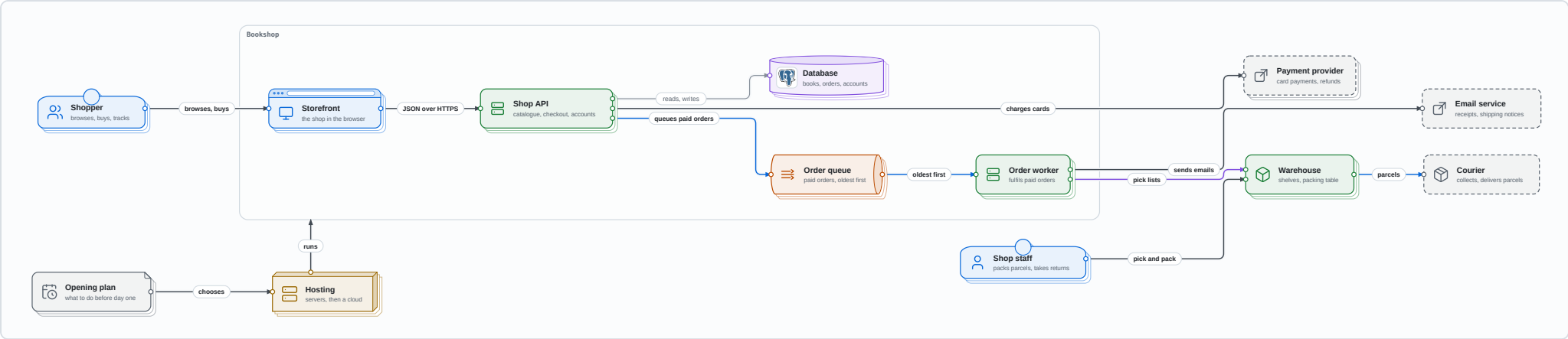
Contents (continued)

- Bookshop schema 32
- Bookshop, level 1 34
  - 1 Take order 35
- Fulfilling an order 36
  - Take payment 38
    - One charge 39
- Checkout 40
  - Checkout: payment accepted 42
- From order to shipment 43
  - Pick and pack 46
- Handling a returned book 47
- Bookshop: production deployment 49
  - Production 50
- Opening the bookshop 52
  - Bookshop on Google Cloud 54
    Icons from a pack you fetch: umlimator.py pack fetch gcp, after Google's terms; glyphs and full names without it. Umlimator is not affiliated with, endorsed by or sponsored by Amazon Web Services, Google or Microsoft.
  - Bookshop on Azure 56
    Icons from a pack you fetch: umlimator.py pack fetch azure, after Microsoft's terms; glyphs and full names without it. Umlimator is not affiliated with, endorsed by or sponsored by Amazon Web Services, Google or Microsoft.
  - Bookshop on AWS 58

Bookshop

13 cards · 12 lines · 1 box

The whole bookshop business. Double-click a card to go in: each opens a diagram of its part, and those open further.



Legend **Client / UI** people and screens · **Service / API** runs the logic · **Queue / Event** hands work on · **Data store** keeps data · **Infra / Cloud** hosts, observability · **External** outside the system · **File / Doc** a document or config · **Person** someone who uses it · **Browser app** runs in a web browser · **⤢ Stacked** opens a view of its own · **calls** runs or invokes it · **feeds** hands work on · **triggers** suggests, offers, fires · **state** reads or writes a file, board, config

Cards 13

Card	Kind	What it is
Shopper ⤢ `browses, buys, tracks`	Client / UI	Someone who buys books from the shop. Opens the system context: who uses the Bookshop and what it uses.
Shop staff ⤢ `packs parcels, takes returns`	Client / UI	The people who run the shop. Opens how they handle a returned book (a flowchart).
Storefront ⤢ `the shop in the browser`	Client / UI	Catalogue, basket and checkout in the browser. Opens buying a book (an interaction overview), whose steps open sequence diagrams.
Shop API ⤢ `catalogue, checkout, accounts`	Service / API	The JSON API behind the storefront. Opens its components, which open the sales domain's classes, an order's life and the order service inside.
Database ⤢ `books, orders, accounts`	Data store	PostgreSQL: books, baskets, orders and accounts. Opens its schema (an ER diagram).

Cards 13 (continued)

Card	Kind	What it is
Order queue ⤢ `paid orders, oldest first`	Queue / Event	Paid orders waiting to be fulfilled. Opens how order data flows through the shop (a data flow diagram, two levels).
Order worker ⤢ `fulfils paid orders`	Service / API	Takes paid orders off the queue and fulfils them. Opens fulfilling an order (an activity diagram), whose payment step opens one charge over time (a timing diagram).
Payment provider ⤢ `card payments, refunds`	External	Takes card payments and makes refunds. Opens checkout (a sequence diagram), and from there the same checkout as a communication diagram.
Warehouse ⤢ `shelves, packing table`	Service / API	Where the books are picked and packed. Opens from order to shipment (BPMN), whose pick and pack opens its own process.
Email service `receipts, shipping notices`	External	Delivers the shop's emails: receipts and shipping notices.
Courier `collects, delivers parcels`	External	Collects parcels at 16:00 and delivers them to shoppers.

Bookshop (continued)

Cards 13 (continued)

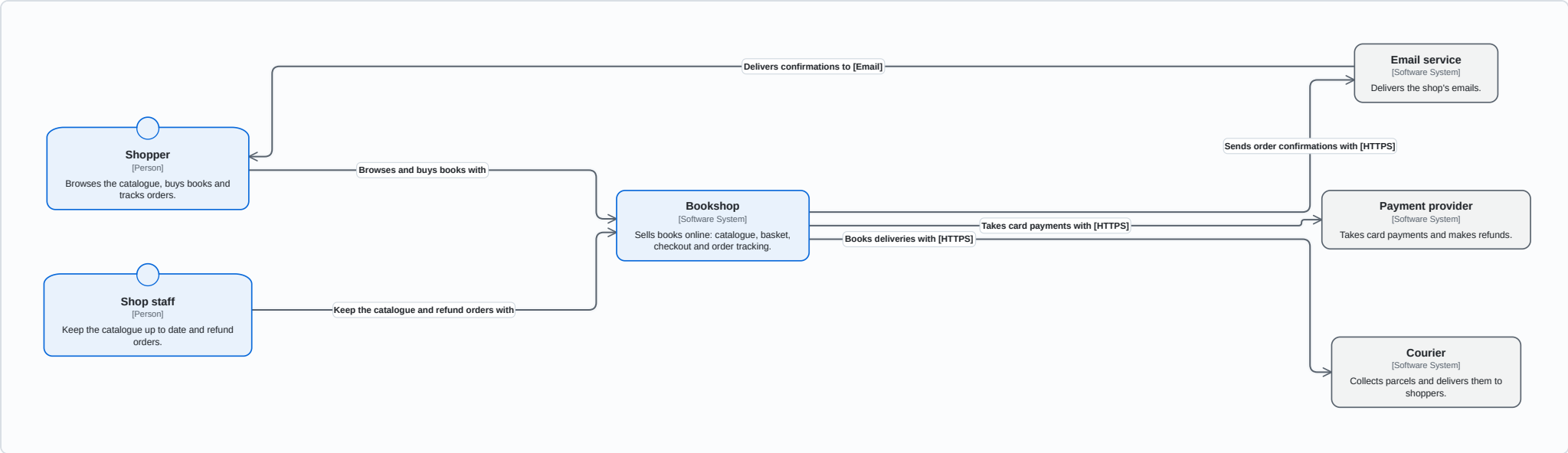
Card	Kind	What it is
Hosting ⤢ `servers, then a cloud`	Infra / Cloud	Where the shop runs. Opens the production deployment (C4), whose Docker host opens the same deployment in UML.
Opening plan ⤢ `what to do before day one`	File / Doc	Everything to do before the shop opens (a mindmap). Its three hosting choices open the shop on Google Cloud, Azure and AWS.

Lines 12

From → to	Label	Technology
Shopper → Storefront	browses, buys	
Storefront → Shop API	JSON over HTTPS	
Shop API → Database	reads, writes	
Shop API → Payment provider	charges cards	
Shop API → Order queue	queues paid orders	
Order queue → Order worker	oldest first	
Order worker → Warehouse	pick lists	
Order worker → Email service	sends emails	
Warehouse → Courier	parcels	
Shop staff → Warehouse	pick and pack	
Hosting → Bookshop	runs	
Opening plan → Hosting	chooses	

Boxes 1

Box	Holds
Bookshop	Storefront, Shop API, Database, Order queue and Order worker



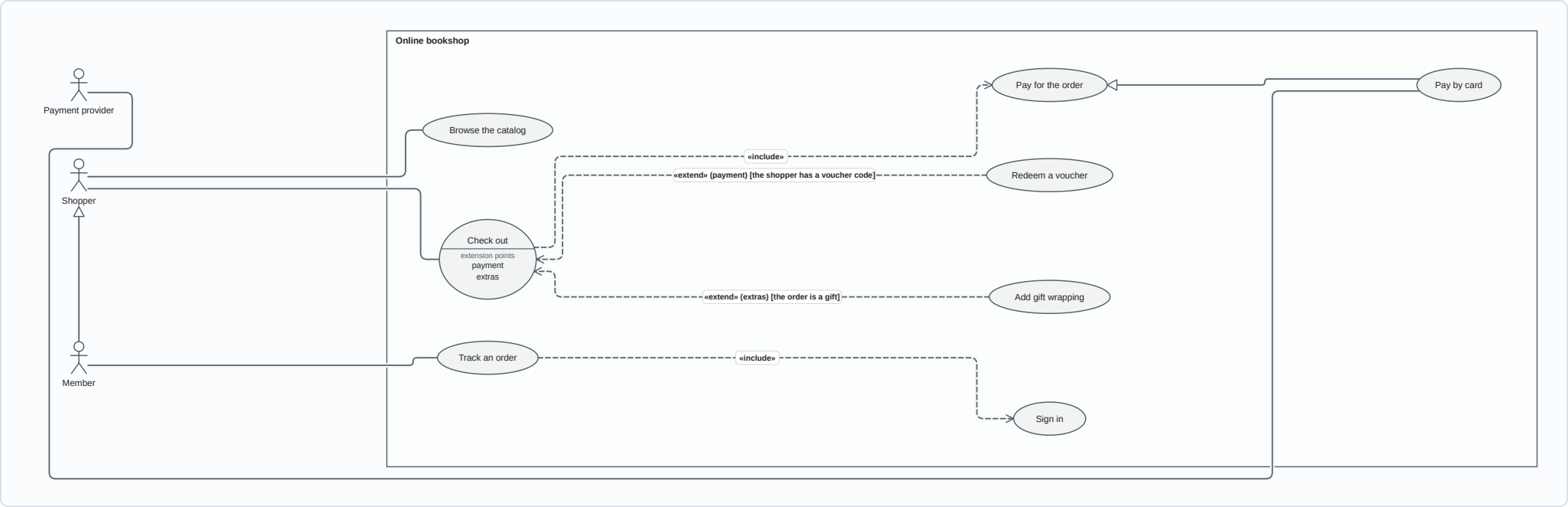
Legend **Person** a person **Software system** a software system **External software system** outside the system, drawn grey **Relationship** uses, with its technology in brackets

Elements 6

Card	Kind	Members
Shopper	Person	Browses the catalogue, buys books and tracks orders.
Shop staff	Person	Keep the catalogue up to date and refund orders.
Bookshop	Software system	Sells books online: catalogue, basket, checkout and order tracking.
Payment provider	Software system	Takes card payments and makes refunds. external
Email service	Software system	Delivers the shop's emails. external
Courier	Software system	Collects parcels and delivers them to shoppers. external

Lines 6

From → to	Label	Kind
Shopper → Bookshop	Browses and buys books with	Relationship
Shop staff → Bookshop	Keep the catalogue and refund orders with	Relationship
Bookshop → Payment provider	Takes card payments with [HTTPS]	Relationship
Bookshop → Email service	Sends order confirmations with [HTTPS]	Relationship
Bookshop → Courier	Books deliveries with [HTTPS]	Relationship
Email service → Shopper	Delivers confirmations to [Email]	Relationship



Legend — **Association** an actor takes part in the use case --> **Include** the base always does the included use case --> **Extend** adds to the base at its extension points, when the condition holds —▷ **Generalization** is a kind of the one it points to

Elements 12

Card	Kind	Members
Shopper	Actor	
Member	Actor	
Payment provider	Actor	
Online bookshop	Subject	
Browse the catalog	Use case	
Check out	Use case	payment extras
Pay for the order	Use case	
Pay by card	Use case	
Redeem a voucher	Use case	

Elements 12 (continued)

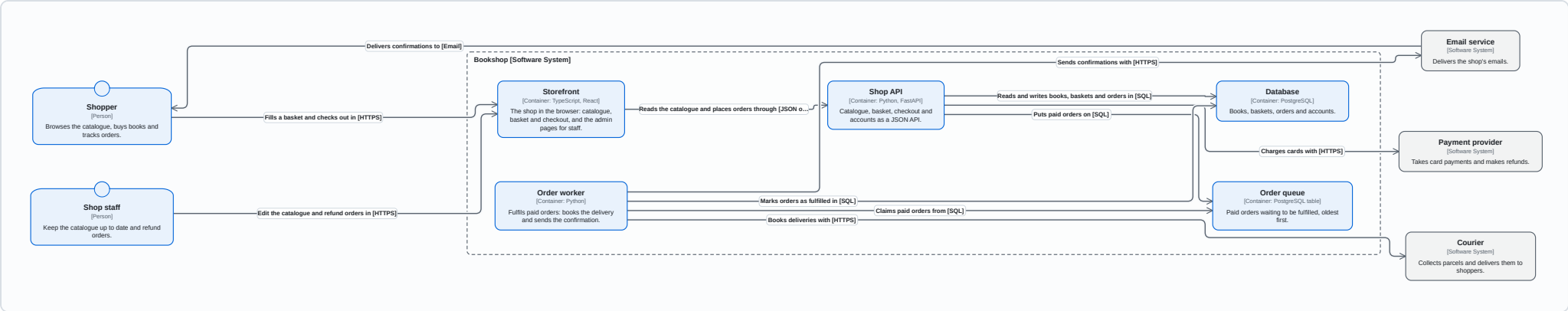
Card	Kind	Members
Add gift wrapping	Use case	
Sign in	Use case	
Track an order	Use case	

Lines 10

From → to	Label	Kind
Member → Shopper		Generalization
Shopper → Browse the catalog		Association
Shopper → Check out		Association
Member → Track an order		Association
Payment provider → Pay by card		Association

Lines 10 (continued)

From → to	Label	Kind
Check out → Pay for the order		Include
Track an order → Sign in		Include
Pay by card → Pay for the order		Generalization
Redeem a voucher → Check out		Extend
Add gift wrapping → Check out		Extend



Legend **Person** a person **Container** a container **External software system** outside the system, drawn grey **Relationship** uses, with its technology in brackets

Elements 11

Card	Kind	Members
Bookshop	Software system	Sells books online: catalogue, basket, checkout and order tracking.
Shopper	Person	Browses the catalogue, buys books and tracks orders.
Shop staff	Person	Keep the catalogue up to date and refund orders.
Storefront	Container	TypeScript, React The shop in the browser: catalogue, basket and checkout, and the admin pages for staff.
Shop API	Container	Python, FastAPI Catalogue, basket, checkout and accounts as a JSON API.
Order worker	Container	Python Fulfils paid orders: books the delivery and sends the confirmation.
Database	Container	PostgreSQL Books, baskets, orders and accounts.
Order queue	Container	PostgreSQL table Paid orders waiting to be fulfilled, oldest first.
Payment provider	Software system	Takes card payments and makes refunds. external

Elements 11 (continued)

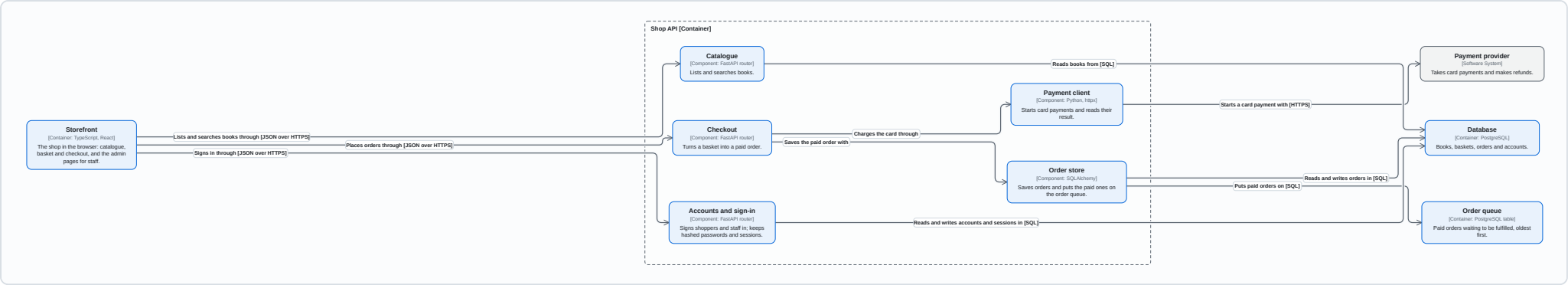
Card	Kind	Members
Email service	Software system	Delivers the shop's emails. external
Courier	Software system	Collects parcels and delivers them to shoppers. external

Lines 11

From → to	Label	Kind
Email service → Shopper	Delivers confirmations to [Email]	Relationship
Shopper → Storefront	Fills a basket and checks out in [HTTPS]	Relationship
Shop staff → Storefront	Edit the catalogue and refund orders in [HTTPS]	Relationship
Storefront → Shop API	Reads the catalogue and places orders through [JSON over HTTPS]	Relationship
Shop API → Payment provider	Charges cards with [HTTPS]	Relationship
Shop API → Database	Reads and writes books, baskets and orders in [SQL]	Relationship
Shop API → Order queue	Puts paid orders on [SQL]	Relationship
Order worker → Order queue	Claims paid orders from [SQL]	Relationship

Lines 11 (continued)

From → to	Label	Kind
Order worker → Courier	Books deliveries with [HTTPS]	Relationship
Order worker → Email service	Sends confirmations with [HTTPS]	Relationship
Order worker → Database	Marks orders as fulfilled in [SQL]	Relationship



Legend **Container** a container **Component** a component **External software system** outside the system, drawn grey **Relationship** uses, with its technology in brackets

Elements 10

Card	Kind	Members
Shop API	Container	Python, FastAPI Catalogue, basket, checkout and accounts as a JSON API.
Storefront	Container	TypeScript, React The shop in the browser: catalogue, basket and checkout, and the admin pages for staff.
Catalogue	Component	FastAPI router Lists and searches books.
Checkout	Component	FastAPI router Turns a basket into a paid order.
Accounts and sign-in	Component	FastAPI router Signs shoppers and staff in; keeps hashed passwords and sessions.
Payment client	Component	Python, httpx Starts card payments and reads their result.
Order store	Component	SQLAlchemy Saves orders and puts the paid ones on the order queue.
Database	Container	PostgreSQL Books, baskets, orders and accounts.

Elements 10 (continued)

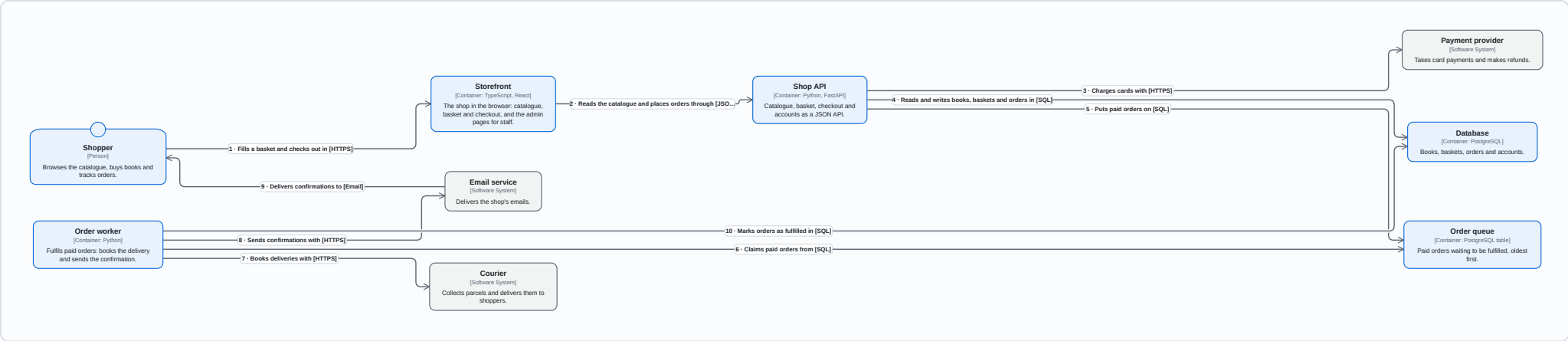
Card	Kind	Members
Order queue	Container	PostgreSQL table Paid orders waiting to be fulfilled, oldest first.
Payment provider	Software system	Takes card payments and makes refunds. external

Lines 10

From → to	Label	Kind
Storefront → Catalogue	Lists and searches books through [JSON over HTTPS]	Relationship
Storefront → Checkout	Places orders through [JSON over HTTPS]	Relationship
Storefront → Accounts and sign-in	Signs in through [JSON over HTTPS]	Relationship
Checkout → Payment client	Charges the card through	Relationship
Payment client → Payment provider	Starts a card payment with [HTTPS]	Relationship
Checkout → Order store	Saves the paid order with	Relationship
Order store → Database	Reads and writes orders in [SQL]	Relationship
Order store → Order queue	Puts paid orders on [SQL]	Relationship

Lines 10 (continued)

From → to	Label	Kind
Catalogue → Database	Reads books from [SQL]	Relationship
Accounts and sign-in → Database	Reads and writes accounts and sessions in [SQL]	Relationship



Legend **Person** a person **Container** a container **External software system** outside the system, drawn grey **Relationship** a step, numbered in time order, with its technology in brackets

Elements 9

Card	Kind	Members
Shopper	Person	Browses the catalogue, buys books and tracks orders.
Storefront	Container	TypeScript, React The shop in the browser: catalogue, basket and checkout, and the admin pages for staff.
Shop API	Container	Python, FastAPI Catalogue, basket, checkout and accounts as a JSON API.
Payment provider	Software system	Takes card payments and makes refunds. external
Database	Container	PostgreSQL Books, baskets, orders and accounts.
Order queue	Container	PostgreSQL table Paid orders waiting to be fulfilled, oldest first.
Order worker	Container	Python Fulfils paid orders: books the delivery and sends the confirmation.

Elements 9 (continued)

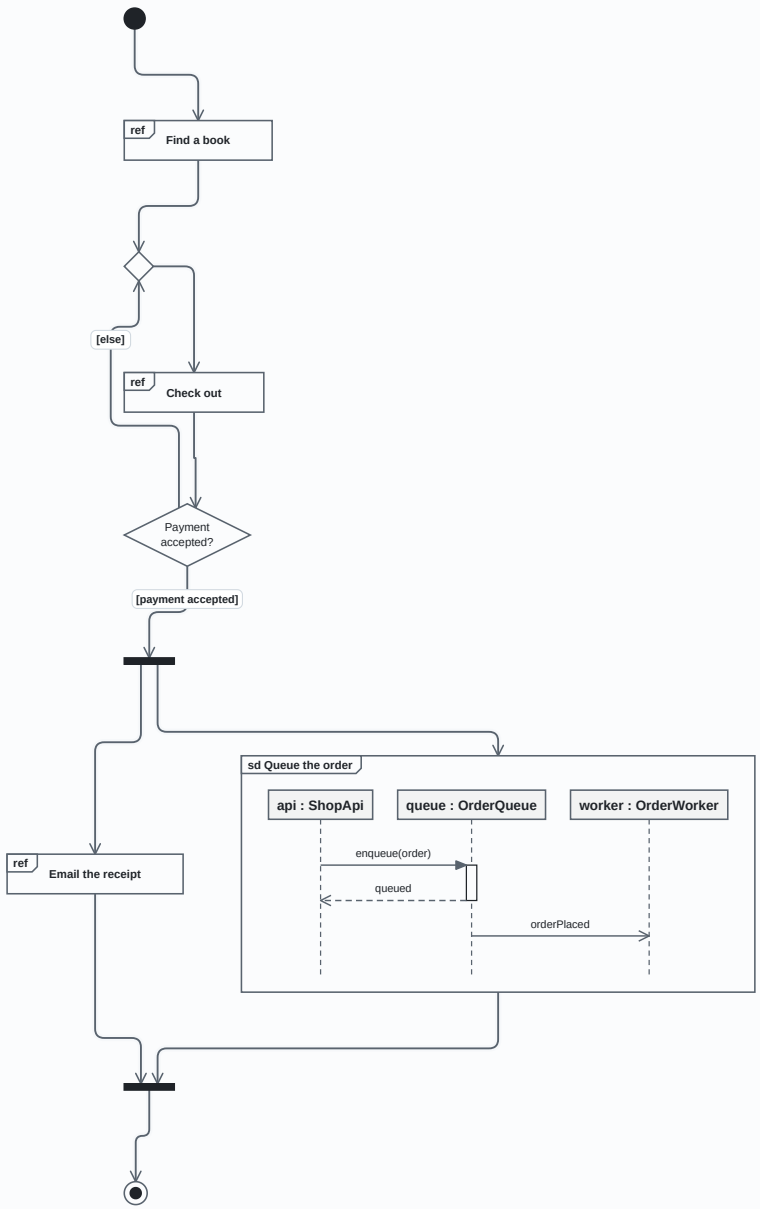
Card	Kind	Members
Courier	Software system	Collects parcels and delivers them to shoppers. external
Email service	Software system	Delivers the shop's emails. external

Lines 10

From → to	Label	Kind
Shopper → Storefront	1 · Fills a basket and checks out in [HTTPS]	Relationship
Storefront → Shop API	2 · Reads the catalogue and places orders through [JSON over HTTPS]	Relationship
Shop API → Payment provider	3 · Charges cards with [HTTPS]	Relationship
Shop API → Database	4 · Reads and writes books, baskets and orders in [SQL]	Relationship
Shop API → Order queue	5 · Puts paid orders on [SQL]	Relationship
Order worker → Order queue	6 · Claims paid orders from [SQL]	Relationship
Order worker → Courier	7 · Books deliveries with [HTTPS]	Relationship

Lines 10 (continued)

From → to	Label	Kind
Order worker → Email service	8 · Sends confirmations with [HTTPS]	Relationship
Email service → Shopper	9 · Delivers confirmations to [Email]	Relationship
Order worker → Database	10 · Marks orders as fulfilled in [SQL]	Relationship



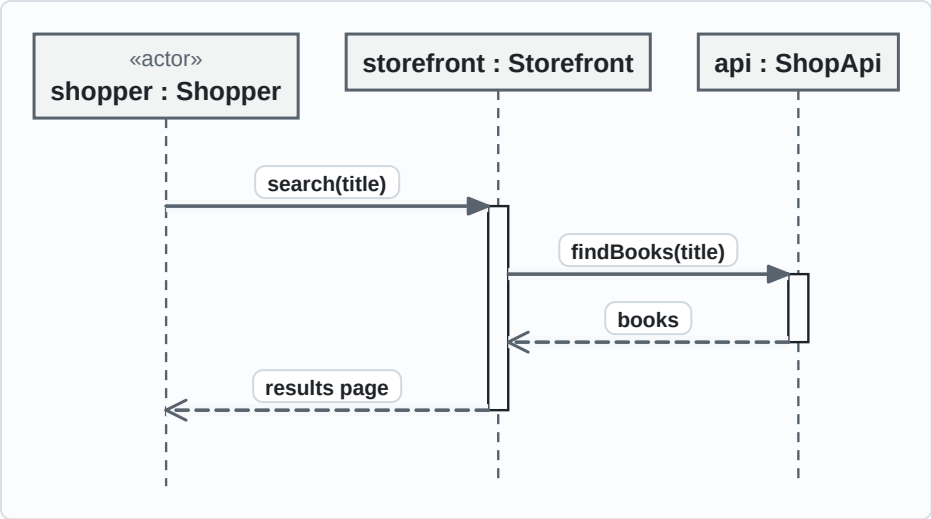
Legend → **Control flow** what runs next · ● **Initial node** where the flow starts · ◉ **Activity final** ends the whole flow · ◇ **Decision** one way out, by its guards · ◇ **Merge** where ways meet · ▌ **Fork** splits into branches that run together · ▌ **Join** waits for all its branches

Elements 10

Card	Kind	Members
Initial node	Initial node	
Find a book	Interaction use	
Merge node	Merge node	
Check out	Interaction use	
Payment accepted?	Decision node	
Fork node	Fork node	
Email the receipt	Interaction use	
Queue the order	Interaction	
Join node	Join node	
Activity final node	Activity final node	

Lines 11

From → to	Label	Kind
Initial node → Find a book		Control flow
Find a book → Merge node		Control flow
Merge node → Check out		Control flow
Check out → Payment accepted?		Control flow
Payment accepted? → Fork node		Control flow
Payment accepted? → Merge node		Control flow
Fork node → Email the receipt		Control flow
Fork node → Queue the order		Control flow
Email the receipt → Join node		Control flow
Queue the order → Join node		Control flow
Join node → Activity final node		Control flow



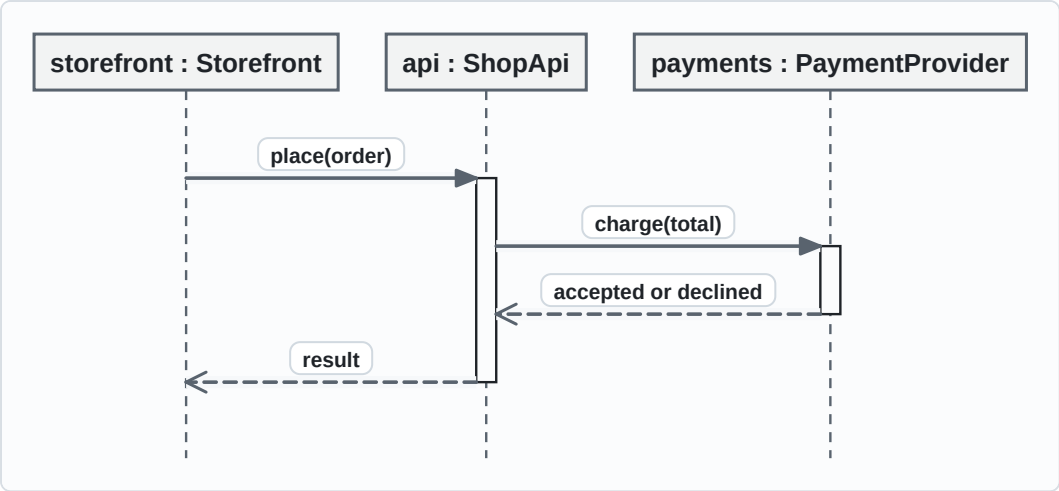
Legend → **Call** the sender waits for it --> **Reply** the answer to a call

Elements 3

Card	Kind	Members
shopper : Shopper	Lifeline	
storefront : Storefront	Lifeline	
api : ShopApi	Lifeline	

Lines 4

From → to	Label	Kind
shopper : Shopper → storefront : Storefront	search(title)	Message
storefront : Storefront → api : ShopApi	findBooks(title)	Message
api : ShopApi → storefront : Storefront	books	Message
storefront : Storefront → shopper : Shopper	results page	Message



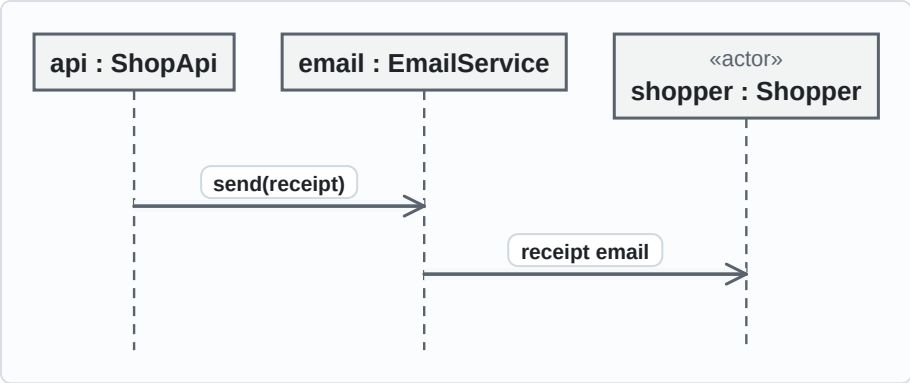
Legend → **Call** the sender waits for it --> **Reply** the answer to a call

Elements 3

Card	Kind	Members
storefront : Storefront	Lifeline	
api : ShopApi	Lifeline	
payments : PaymentProvider	Lifeline	

Lines 4

From → to	Label	Kind
storefront : Storefront → api : ShopApi	place(order)	Message
api : ShopApi → payments : PaymentProvider	charge(total)	Message
payments : PaymentProvider → api : ShopApi	accepted or declined	Message
api : ShopApi → storefront : Storefront	result	Message



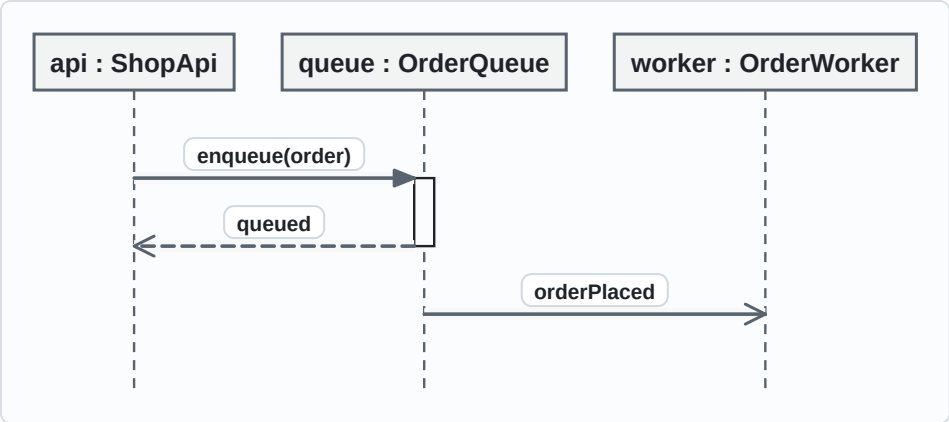
Legend —> **Async** the sender goes on

Elements 3

Card	Kind	Members
api : ShopApi	Lifeline	
email : EmailService	Lifeline	
shopper : Shopper	Lifeline	

Lines 2

From → to	Label	Kind
api : ShopApi → email : EmailService	send(receipt)	Message
email : EmailService → shopper : Shopper	receipt email	Message



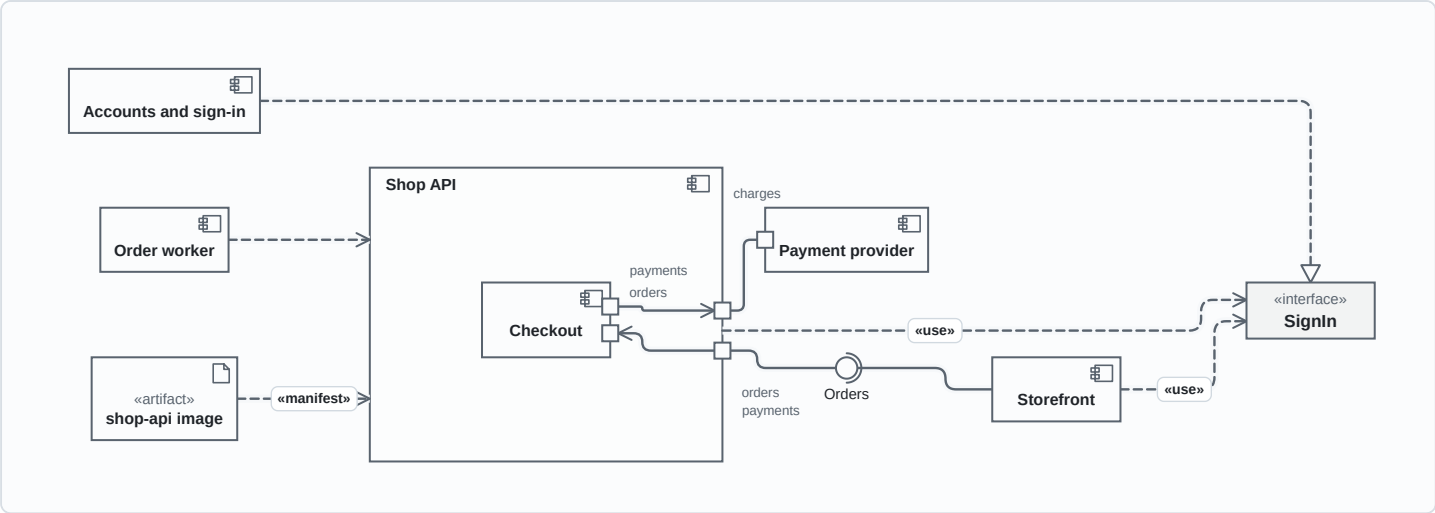
Legend → **Call** the sender waits for it —> **Async** the sender goes on
--> **Reply** the answer to a call

Elements 3

Card	Kind	Members
api : ShopApi	Lifeline	
queue : OrderQueue	Lifeline	
worker : OrderWorker	Lifeline	

Lines 3

From → to	Label	Kind
api : ShopApi → queue : OrderQueue	enqueue(order)	Message
queue : OrderQueue → api : ShopApi	queued	Message
queue : OrderQueue → worker : OrderWorker	orderPlaced	Message



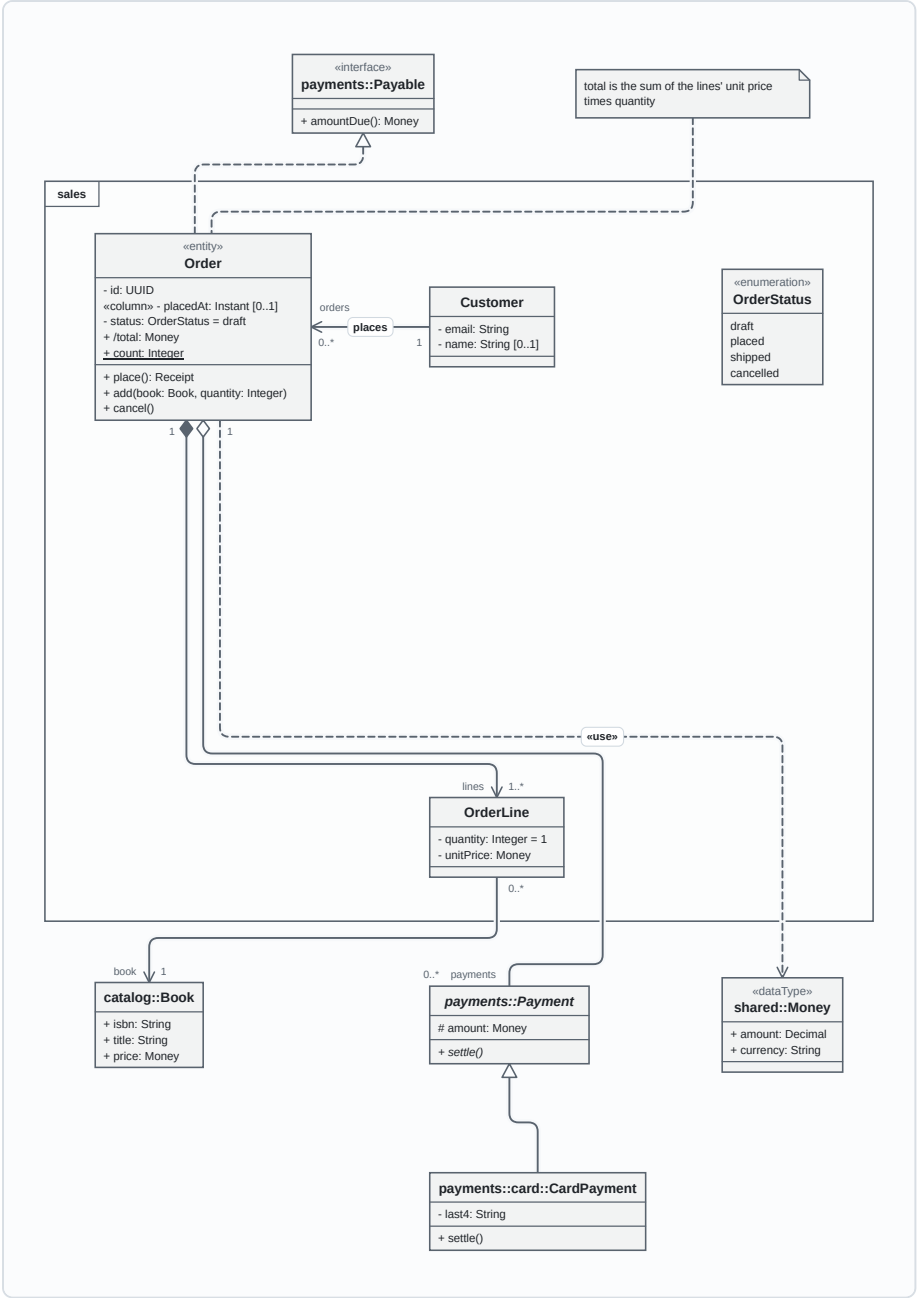
Legend —○ **Realization** provides the interface drawn as a ball (its stem) —(**Usage** requires the interface: a socket round its ball --▷ **Realization** provides the interface it points to
--> **Usage** requires the one it points to —— **Assembly** joins what one part requires to what another provides ——> **Delegation** passes on, inward or out, what a port takes or needs
--> **Dependency** uses the one it points to --> **Manifestation** the artifact is the component's file

Elements 9

Card	Kind	Members
Storefront	Component	
Shop API	Component	orders, provides Orders payments
Checkout	Component	orders payments
Accounts and sign-in	Component	
Payment provider	Component	charges
Order worker	Component	
Orders	Interface	
SignIn	Interface	
shop-api image	Artifact	manifests Shop API

Lines 9

From → to	Label	Kind
Storefront → Orders		Usage
Accounts and sign-in → SignIn		Interface realization
Storefront → SignIn		Usage
Shop API → SignIn		Usage
Order worker → Shop API		Dependency
shop-api image → Shop API		Manifestation
Shop API → Checkout	Delegation from orders of Shop API to orders of Checkout	Connector
Checkout → Shop API	Delegation from payments of Checkout to payments of Shop API	Connector
Shop API → Payment provider	Assembly from payments of Shop API to charges of Payment provider	Connector



Legend — **Association** a link between instances — **Generalization** is a kind of the one it points to

--▷ **Realization** implements the interface it points to --> **Dependency** uses the one it points to

---- **Comment link** what a comment is about —◆ **Composition** the whole owns its parts

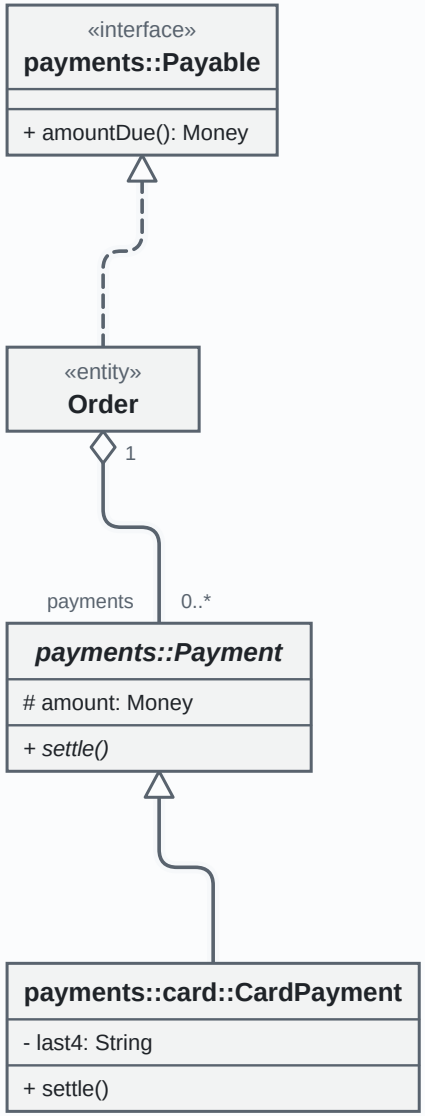
—◇ **Aggregation** the whole holds its parts —> **Navigable end** reachable from the other end

Elements 11

Card	Kind	Members
sales	Package	
Customer	Class	- email: String - name: String [0..1]
Order	Class	- id: UUID «column» - placedAt: Instant [0..1] - status: OrderStatus = draft + /total: Money + {static} count: Integer + place(): Receipt + add(book: Book, quantity: Integer) + cancel()
OrderLine	Class	- quantity: Integer = 1 - unitPrice: Money
catalog::Book	Class	+ isbn: String + title: String + price: Money
payments::Payable	Interface	+ amountDue(): Money
payments::Payment	Class	# amount: Money + {abstract} settle()
payments::card::CardPayment	Class	- last4: String + settle()
OrderStatus	Enumeration	draft placed shipped cancelled
shared::Money	Data type	+ amount: Decimal + currency: String
total is the sum of the lines' unit price times quantity	Comment	

Lines 8

From → to	Label	Kind
Customer → Order	places	Association
Order → OrderLine		Association
OrderLine → catalog::Book		Association
Order → payments::Payment		Association
Order → payments::Payable		Interface realization
payments::card::CardPayment → payments::Payment		Generalization
Order → shared::Money		Dependency
total is the sum of the lines' unit price times quantity → Order		Comment link



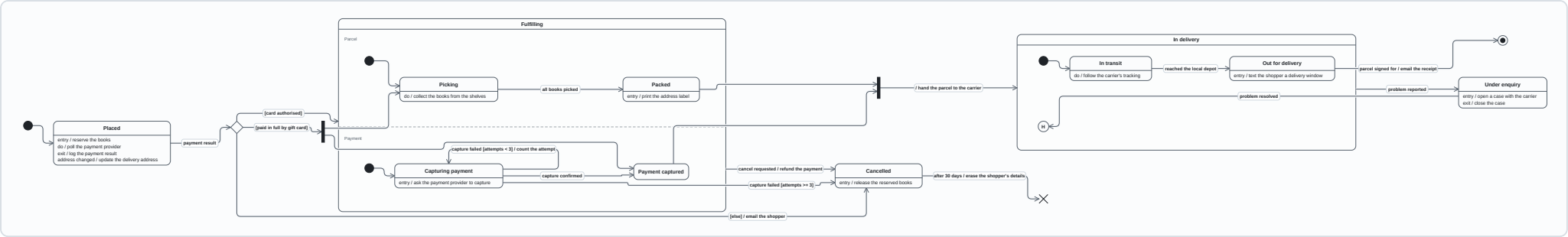
Legend — **Association** a link between instances → **Generalization** is a kind of the one it points to
--▷ **Realization** implements the interface it points to ◇— **Aggregation** the whole holds its parts

Elements 4

Card	Kind	Members
Order	Class	- id: UUID «column» - placedAt: Instant [0..1] - status: OrderStatus = draft + /total: Money + {static} count: Integer + place(): Receipt + add(book: Book, quantity: Integer) + cancel()
payments::Payable	Interface	+ amountDue(): Money
payments::Payment	Class	# amount: Money + {abstract} settle()
payments::card::CardPayment	Class	- last4: String + settle()

Lines 3

From → to	Label	Kind
Order → payments::Payment		Association
Order → payments::Payable		Interface realization
payments::card::CardPayment → payments::Payment		Generalization



Legend → **Transition** trigger [guard] / effect: from one state to the next ● **Initial** where a region starts ◉ **Final state** where a region ends ◇ **Choice** one exit, by the guards

Fork or join into or out of regions at once Ⓗ **History** H back to the last substate, H* at any depth ✕ **Terminate** the machine stops

Elements 21

Card	Kind	Members
Initial pseudostate	Initial pseudostate	
Placed	State	entry / reserve the books do / poll the payment provider exit / log the payment result address changed / update the delivery address
Choice	Choice	
Fork	Fork	
Fulfilling	Composite state	region Parcel region Payment
Initial pseudostate	Initial pseudostate	
Picking	State	do / collect the books from the shelves
Packed	State	entry / print the address label
Initial pseudostate	Initial pseudostate	
Capturing payment	State	entry / ask the payment provider to capture
Payment captured	State	
Join	Join	
In delivery	Composite state	
Initial pseudostate	Initial pseudostate	
In transit	State	do / follow the carrier's tracking

Elements 21 (continued)

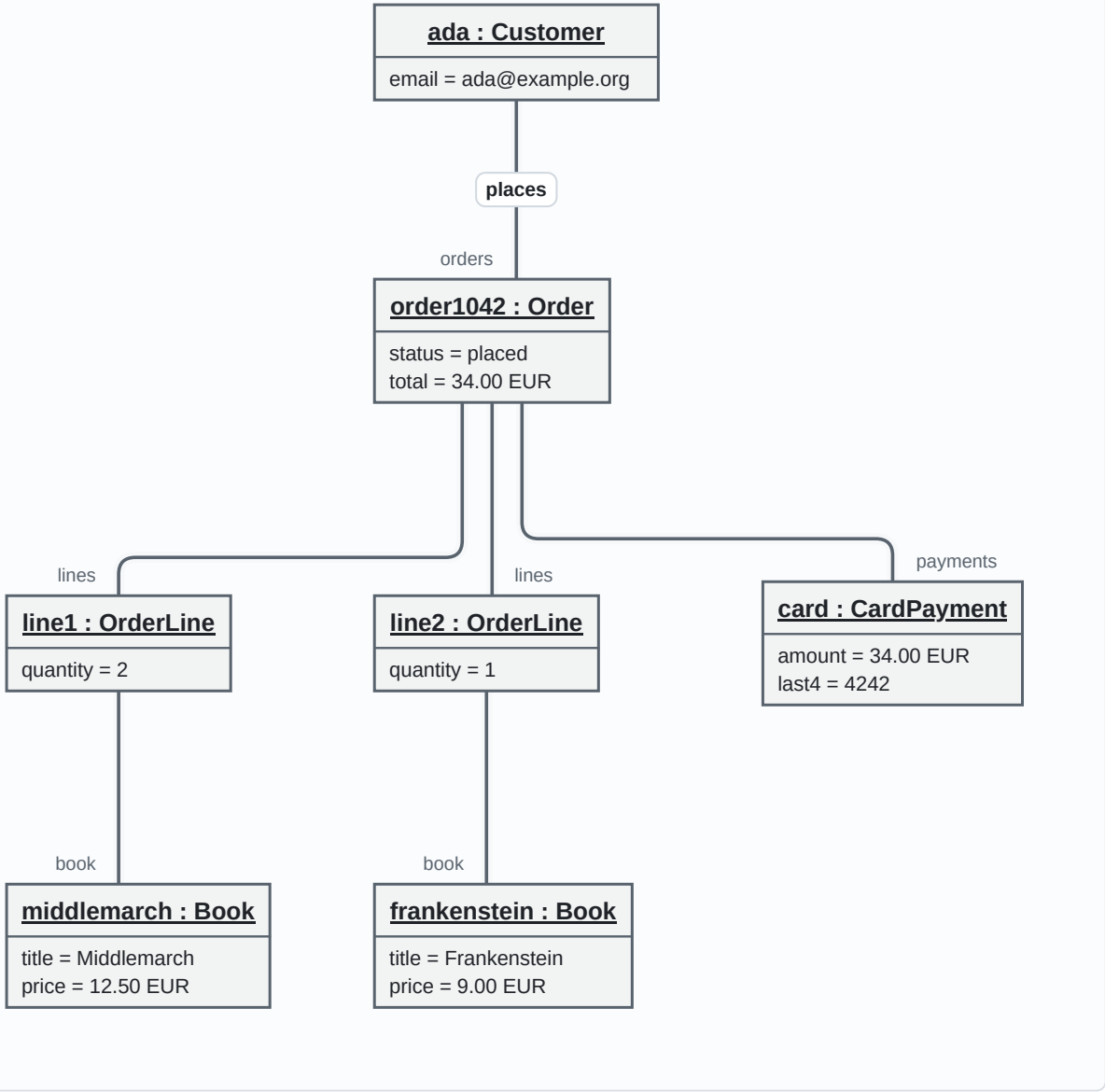
Card	Kind	Members
Out for delivery	State	entry / text the shopper a delivery window
Shallow history	Shallow history	
Under enquiry	State	entry / open a case with the carrier exit / close the case
Cancelled	State	entry / release the reserved books
Terminate	Terminate	
Final state	Final state	

Lines 23

From → to	Label	Kind
Initial pseudostate → Placed		Transition
Placed → Choice		Transition
Choice → Fulfilling		Transition
Choice → Fork		Transition
Choice → Cancelled		Transition
Fork → Picking		Transition
Fork → Payment captured		Transition
Initial pseudostate → Picking		Transition
Picking → Packed		Transition

Lines 23 (continued)

From → to	Label	Kind
Initial pseudostate → Capturing payment		Transition
Capturing payment → Capturing payment		Transition
Capturing payment → Cancelled		Transition
Capturing payment → Payment captured		Transition
Packed → Join		Transition
Payment captured → Join		Transition
Join → In delivery		Transition
Initial pseudostate → In transit		Transition
In transit → Out for delivery		Transition
Out for delivery → Final state		Transition
In delivery → Under enquiry		Transition
Under enquiry → Shallow history		Transition
Fulfilling → Cancelled		Transition
Cancelled → Terminate		Transition



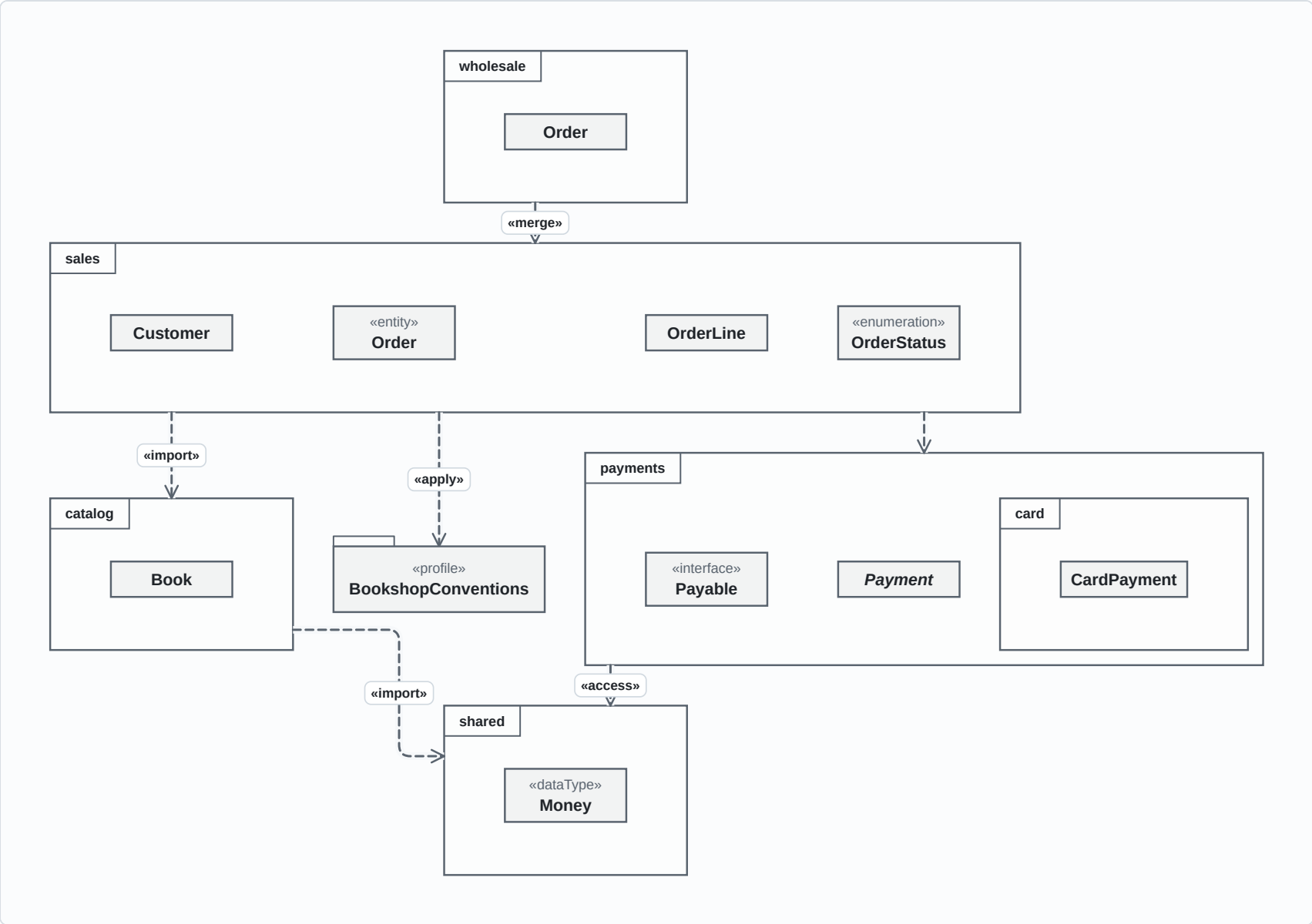
Legend — **Link** joins two instances, an instance of an association

Elements 7

Card	Kind	Members
ada : Customer	Instance specification	email = ada@example.org
order1042 : Order	Instance specification	status = placed total = 34.00 EUR
line1 : OrderLine	Instance specification	quantity = 2
line2 : OrderLine	Instance specification	quantity = 1
middlemarch : Book	Instance specification	title = Middlemarch price = 12.50 EUR
frankenstein : Book	Instance specification	title = Frankenstein price = 9.00 EUR
card : CardPayment	Instance specification	amount = 34.00 EUR last4 = 4242

Lines 6

From → to	Label	Kind
— ada : Customer → order1042 : Order	places	Link
— order1042 : Order → line1 : OrderLine		Link
— order1042 : Order → line2 : OrderLine		Link
— line1 : OrderLine → middlemarch : Book		Link
— line2 : OrderLine → frankenstein : Book		Link
— order1042 : Order → card : CardPayment		Link



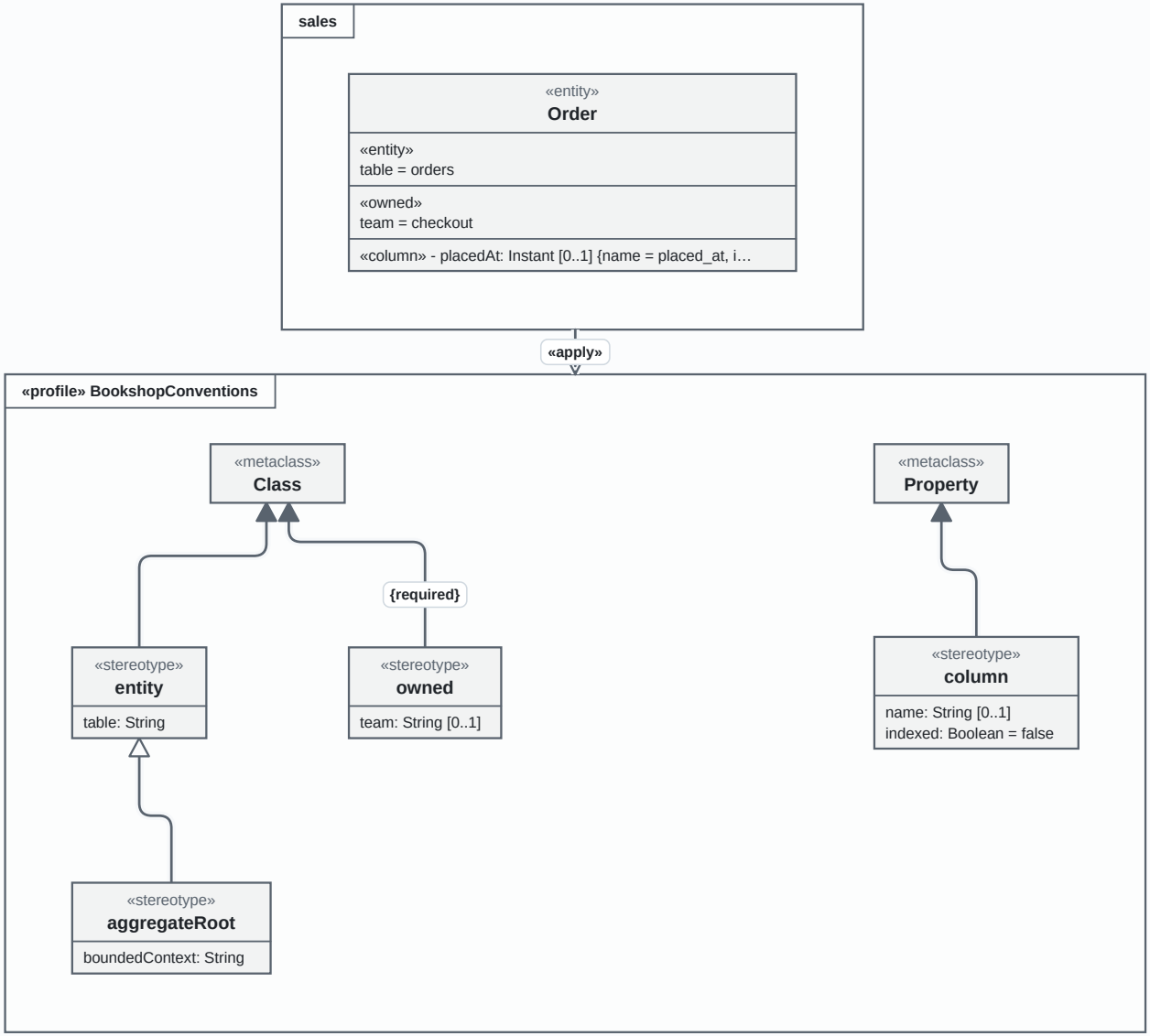
Legend --> **Dependency** uses the one it points to --> **Package import** sees the names of the one it points to: «import», or «access» kept private --> **Package merge** takes in what the one it points to holds --> **Profile application** «apply»: the package may use the profile's stereotypes

Elements 17

Card	Kind	Members
shared	Package	
Money	Data type	
catalog	Package	
Book	Class	
sales	Package	
Customer	Class	
Order	Class	
OrderLine	Class	
OrderStatus	Enumeration	
payments	Package	
Payable	Interface	
Payment	Class	
card	Package	
CardPayment	Class	
wholesale	Package	
Order	Class	
BookshopConventions	Profile	

Lines 6

From → to	Label	Kind
catalog → shared		Package import
sales → catalog		Package import
sales → BookshopConventions		Profile application
payments → shared		Package import
wholesale → sales		Package merge
sales → payments		Dependency



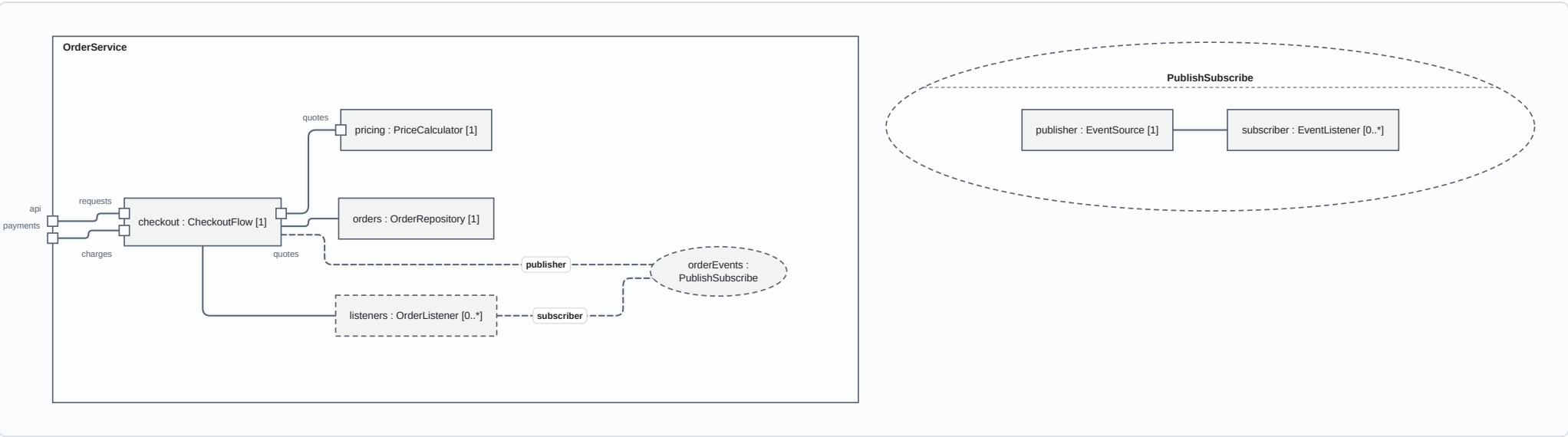
Legend → **Generalization** is a kind of the one it points to
→ **Extension** the stereotype extends the metaclass it points to
--> **Profile application** «apply»: the package may use the profile's s

Elements 9

Card	Kind	Members
BookshopConventions	Profile	metaclass Class metaclass Property «entity» «aggregateRoot» «owned» «column» applied to sales
Class	Metaclass	extended by «entity» extended by «owned»
Property	Metaclass	extended by «column»
entity	Stereotype	extends Class table: String
aggregateRoot	Stereotype	a kind of «entity» boundedContext: String
owned	Stereotype	extends Class {required} team: String [0..1]
column	Stereotype	extends Property name: String [0..1] indexed: Boolean = false
sales	Package	
Order	Class	«entity» table = orders «owned» required team = checkout - placedAt: Instant [0..1]: «column» name = placed_at, indexed = true

Lines 5

From → to	Label	Kind
sales → BookshopConventions		Profile application
entity → Class		Extension
owned → Class		Extension
column → Property		Extension
aggregateRoot → entity		Generalization



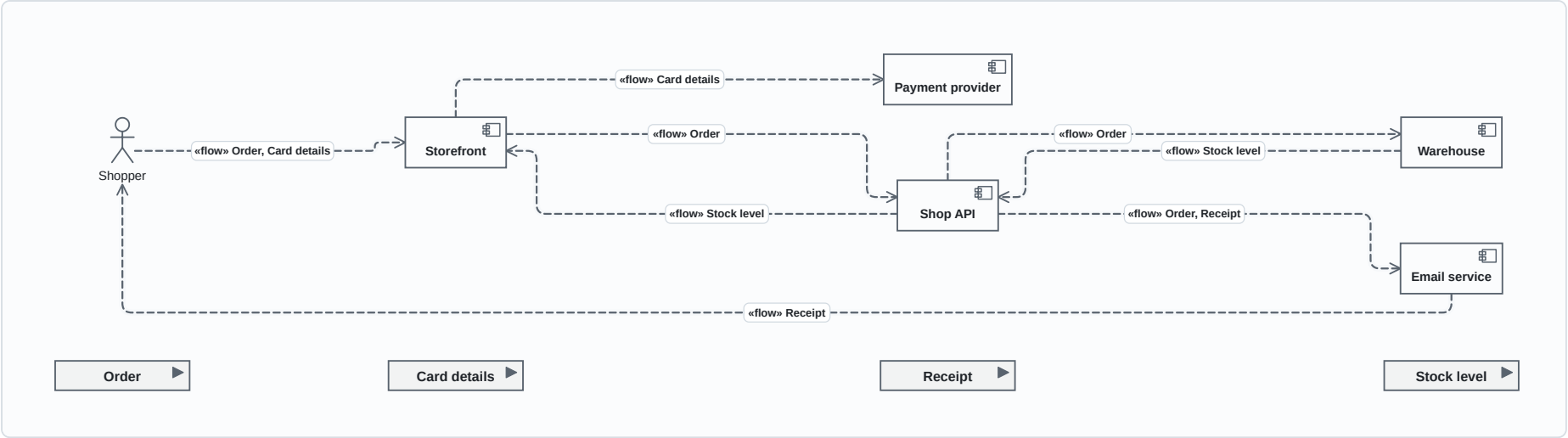
Legend —— **Connector** a link between parts and ports at run time ---- **Role binding** the part plays the named role of the collaboration

Elements 3

Card	Kind	Members
OrderService	Class	port api port payments checkout : CheckoutFlow [1], composite pricing : PriceCalculator [1], composite orders : OrderRepository [1], composite listeners : OrderListener [0..*], shared
orderEvents : PublishSubscribe	Collaboration use	PublishSubscribe publisher ← checkout subscriber ← listeners
PublishSubscribe	Collaboration	publisher : EventSource [1] subscriber : EventListener [0..*]

Lines 8

From → to	Label	Kind
—— OrderService → checkout : CheckoutFlow [1]		Connector
—— checkout : CheckoutFlow [1] → pricing : PriceCalculator [1]		Connector
—— checkout : CheckoutFlow [1] → OrderService		Connector
—— checkout : CheckoutFlow [1] → orders : OrderRepository [1]		Connector
—— checkout : CheckoutFlow [1] → listeners : OrderListener [0..*]		Connector
—— publisher : EventSource [1] → subscriber : EventListener [0..*]		Connector
—— checkout : CheckoutFlow [1] → orderEvents : PublishSubscribe	publisher	Role binding
—— listeners : OrderListener [0..*] → orderEvents : PublishSubscribe	subscriber	Role binding



Legend --> **Information flow** «flow»: what travels from one end to the other

Elements 10

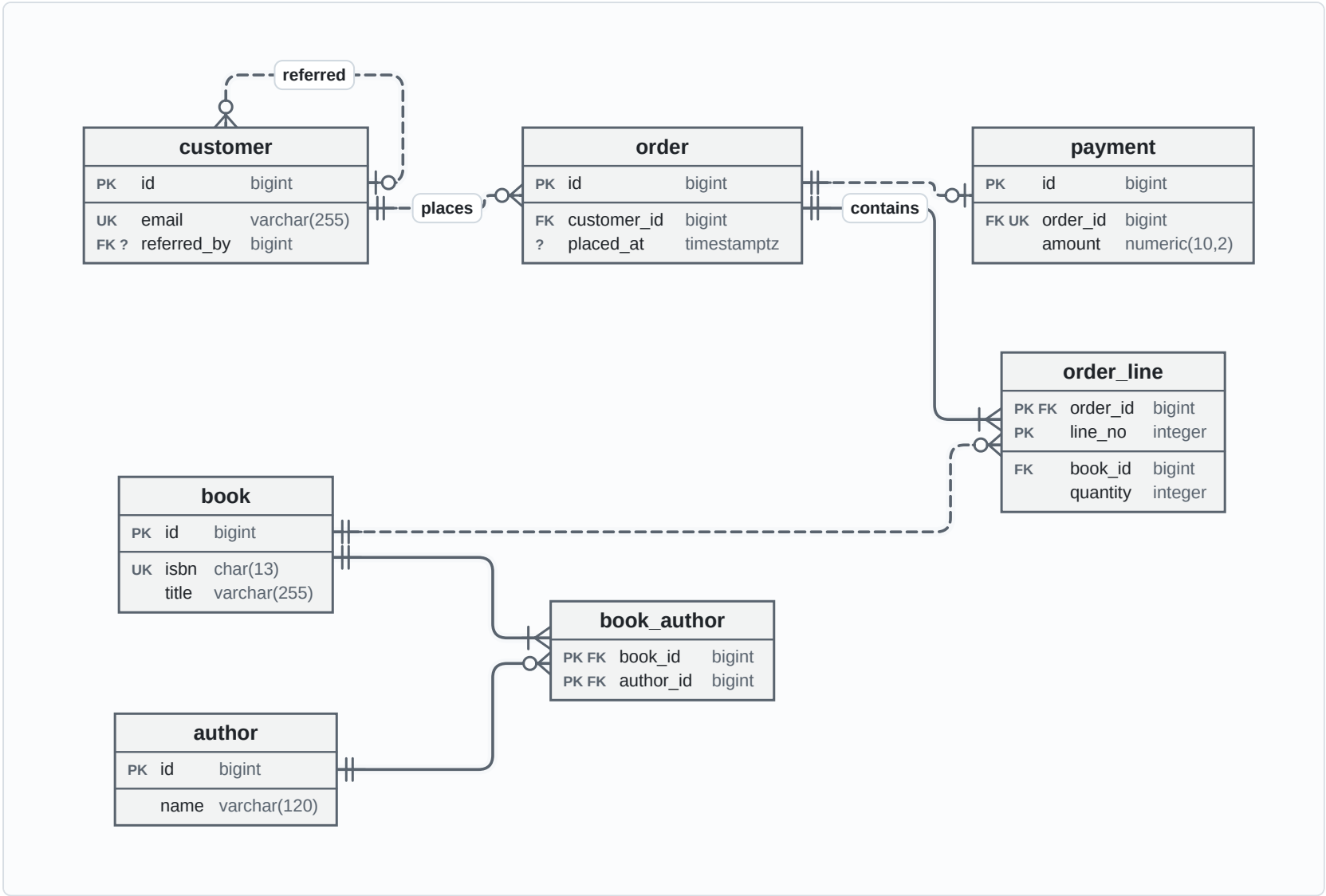
Card	Kind	Members
Shopper	Actor	
Storefront	Component	
Shop API	Component	orders payments
Payment provider	Component	charges
Warehouse	Component	
Email service	Component	
Order	Information item	Shopper → Storefront Storefront → Shop API Shop API → Warehouse Shop API → Email service
Card details	Information item	Shopper → Storefront Storefront → Payment provider

Elements 10 (continued)

Card	Kind	Members
Receipt	Information item	Shop API → Email service Email service → Shopper
Stock level	Information item	Warehouse → Shop API Shop API → Storefront

Lines 8

From → to	Label	Kind
Shopper → Storefront	Order, Card details	Information flow
Storefront → Payment provider	Card details	Information flow
Storefront → Shop API	Order	Information flow
Shop API → Warehouse	Order	Information flow
Warehouse → Shop API	Stock level	Information flow
Shop API → Storefront	Stock level	Information flow
Shop API → Email service	Order, Receipt	Information flow
Email service → Shopper	Receipt	Information flow



Legend — **Identifying** the child's key holds the parent's key ---- **Non-identifying** the child refers to the parent outside its key —|| **Exactly one** one and only one at that end

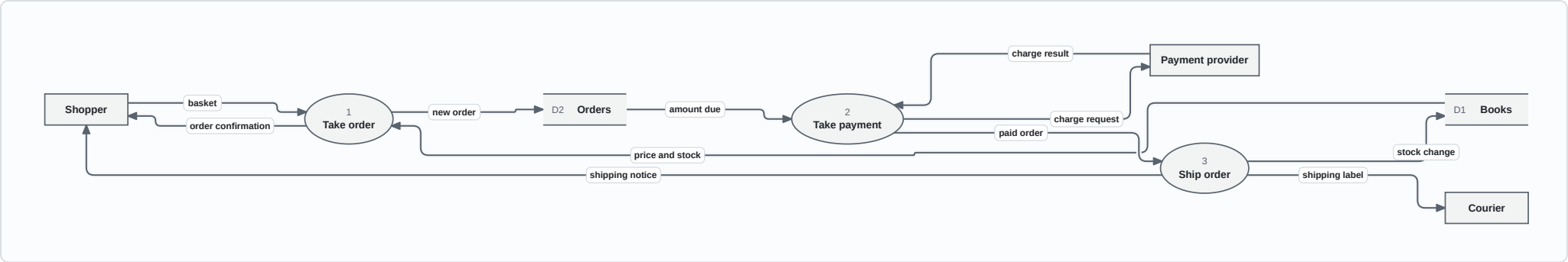
—o| **Zero or one** at most one at that end —|< **One or more** at least one at that end —o< **Zero or more** any number at that end

Elements 7

Card	Kind	Members
customer	Table	id bigint PK email varchar(255) UK referred_by bigint FK customer.id NULL
order	Table	id bigint PK customer_id bigint FK customer.id placed_at timestamptz NULL
order_line	Table	order_id bigint PK FK order.id line_no integer PK book_id bigint FK book.id quantity integer
book	Table	id bigint PK isbn char(13) UK title varchar(255)
author	Table	id bigint PK name varchar(120)
book_author	Table	book_id bigint PK FK book.id author_id bigint PK FK author.id
payment	Table	id bigint PK order_id bigint FK order.id UK amount numeric(10,2)

Lines 7

From → to	Label	Kind
customer → order	places: 1 to 0..*, non-identifying	Relationship
order → order_line	contains: 1 to 1..*, identifying	Relationship
book → order_line	1 to 0..*, non-identifying	Relationship
book → book_author	1 to 1..*, identifying	Relationship
author → book_author	1 to 0..*, identifying	Relationship
order → payment	1 to 0..1, non-identifying	Relationship
customer → customer	referred: 0..1 to 0..*, non-identifying	Relationship



Legend **Process** transforms the data flowing in into the data flowing out **Data store** holds data at rest **External entity** outside the system: a source or a sink of data

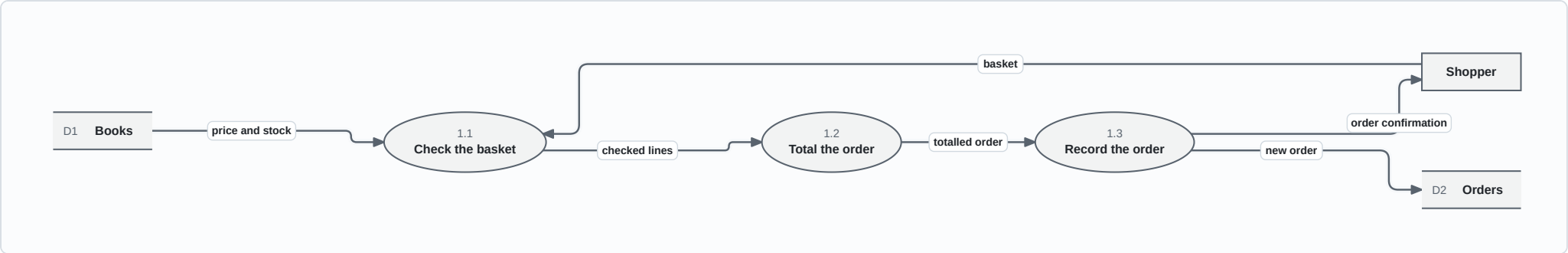
Data flow data moving, named by what it carries

Elements 8

Card	Kind	Members
Shopper	External entity	
Payment provider	External entity	
Courier	External entity	
Take order	Process	number 1 opens level 1 Take order
Take payment	Process	number 2
Ship order	Process	number 3
Books	Data store	number D1
Orders	Data store	number D2

Lines 11

From → to	Label	Kind
Shopper → Take order	basket	Data flow
Books → Take order	price and stock	Data flow
Take order → Orders	new order	Data flow
Take order → Shopper	order confirmation	Data flow
Orders → Take payment	amount due	Data flow
Take payment → Payment provider	charge request	Data flow
Payment provider → Take payment	charge result	Data flow
Take payment → Ship order	paid order	Data flow
Ship order → Books	stock change	Data flow
Ship order → Courier	shipping label	Data flow
Ship order → Shopper	shipping notice	Data flow



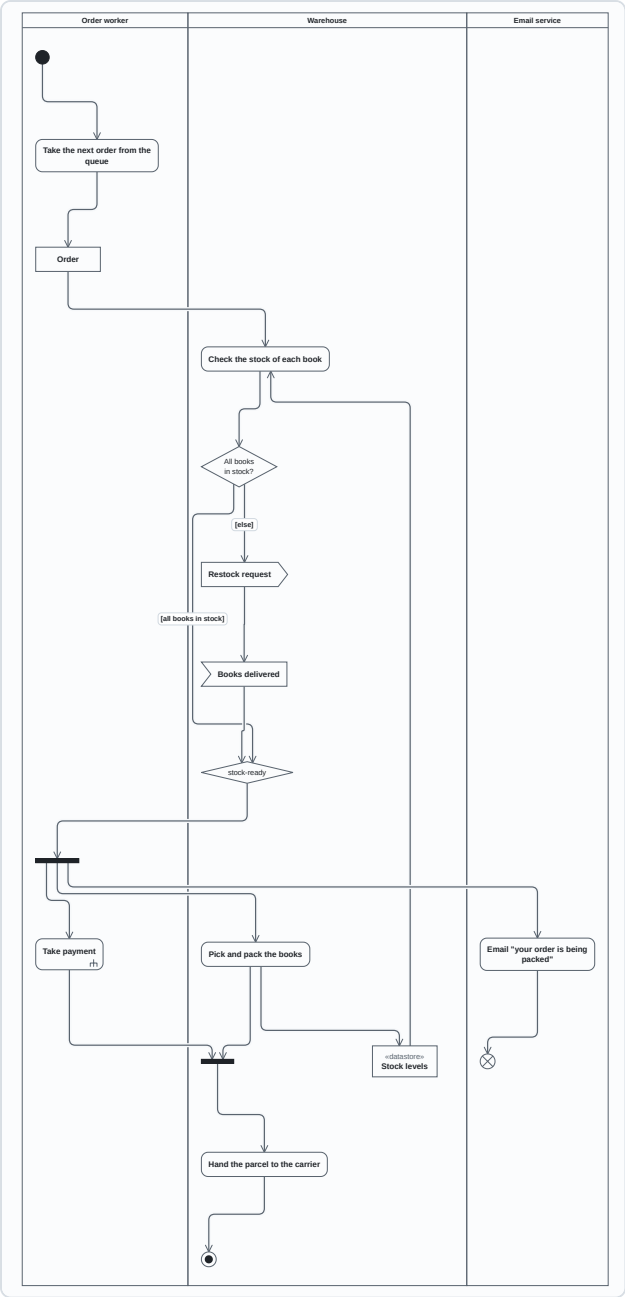
Legend **Process** transforms the data flowing in into the data flowing out **Data store** holds data at rest **External entity** outside the system: a source or a sink of data **Data flow** data moving, named by what it carries

Elements 6

Card	Kind	Members
Shopper	External entity	
Books	Data store	number D1
Orders	Data store	number D2
Check the basket	Process	number 1.1
Total the order	Process	number 1.2
Record the order	Process	number 1.3

Lines 6

From → to	Label	Kind
Shopper → Check the basket	basket	Data flow
Books → Check the basket	price and stock	Data flow
Check the basket → Total the order	checked lines	Data flow
Total the order → Record the order	totalled order	Data flow
Record the order → Orders	new order	Data flow
Record the order → Shopper	order confirmation	Data flow



Legend → **Control flow** the next step, once the one before ends → **Object flow** an object or data passed on

● **Initial node** where the activity starts ◉ **Activity final** ends the whole activity ⊗ **Flow final** ends its own branch only

| **Fork** splits the flow into branches that run at once | **Join** waits for every branch, then goes on

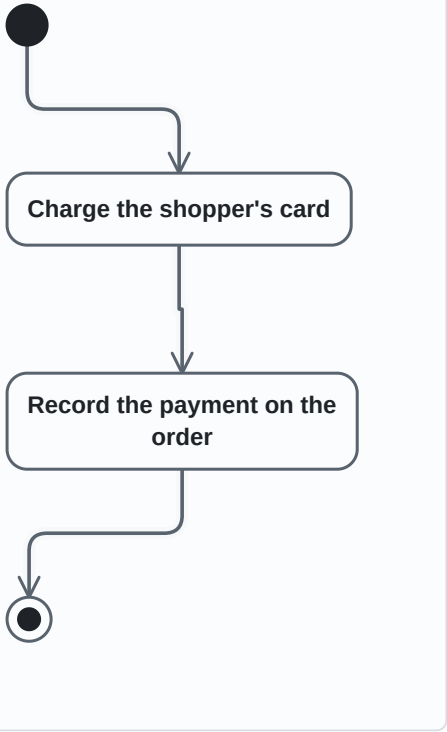
◇ **Decision** takes the one exit whose guard holds ◇ **Merge** goes on with whichever flow arrives

Elements 20

Card	Kind	Members
Order worker	Partition	
Warehouse	Partition	
Email service	Partition	
Initial node	Initial node	in Order worker
Take the next order from the queue	Action	in Order worker
Order	Object node	in Order worker
Check the stock of each book	Action	in Warehouse
Stock levels	Data store	in Warehouse
All books in stock?	Decision	in Warehouse
Restock request	Send signal action	in Warehouse
Books delivered	Accept event action	in Warehouse
stock-ready	Merge	in Warehouse
Fork	Fork	in Order worker
Take payment	Call behaviour action	in Order worker
Pick and pack the books	Action	in Warehouse
Email "your order is being packed"	Action	in Email service
Flow final	Flow final	in Email service
Join	Join	in Warehouse
Hand the parcel to the carrier	Action	in Warehouse
Activity final	Activity final	in Warehouse

Lines 19

From → to	Label	Kind
— Initial node → Take the next order from the queue		Control flow
— Take the next order from the queue → Order		Object flow
— Order → Check the stock of each book		Object flow
— Stock levels → Check the stock of each book		Object flow
— Check the stock of each book → All books in stock?		Control flow
— All books in stock? → stock-ready		Control flow
— All books in stock? → Restock request		Control flow
— Restock request → Books delivered		Control flow
— Books delivered → stock-ready		Control flow
— stock-ready → Fork		Control flow
— Fork → Take payment		Control flow
— Fork → Pick and pack the books		Control flow
— Fork → Email "your order is being packed"		Control flow
— Email "your order is being packed" → Flow final		Control flow
— Pick and pack the books → Stock levels		Object flow
— Take payment → Join		Control flow
— Pick and pack the books → Join		Control flow
— Join → Hand the parcel to the carrier		Control flow
— Hand the parcel to the carrier → Activity final		Control flow



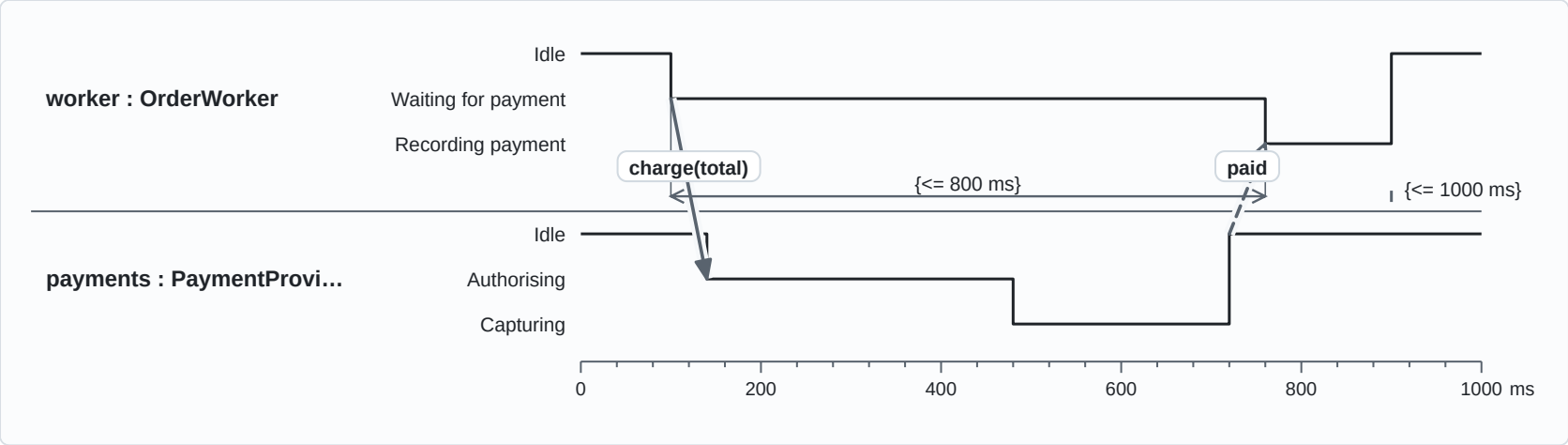
Legend → **Control flow** the next step, once the one before ends ● **Initial node** where the activity starts ◉ **Activity final** ends the whole activity

Elements 4

Card	Kind	Members
Initial node	Initial node	
Charge the shopper's card	Action	
Record the payment on the order	Action	
Activity final	Activity final	

Lines 3

From → to	Label	Kind
Initial node → Charge the shopper's card		Control flow
Charge the shopper's card → Record the payment on the order		Control flow
Record the payment on the order → Activity final		Control flow



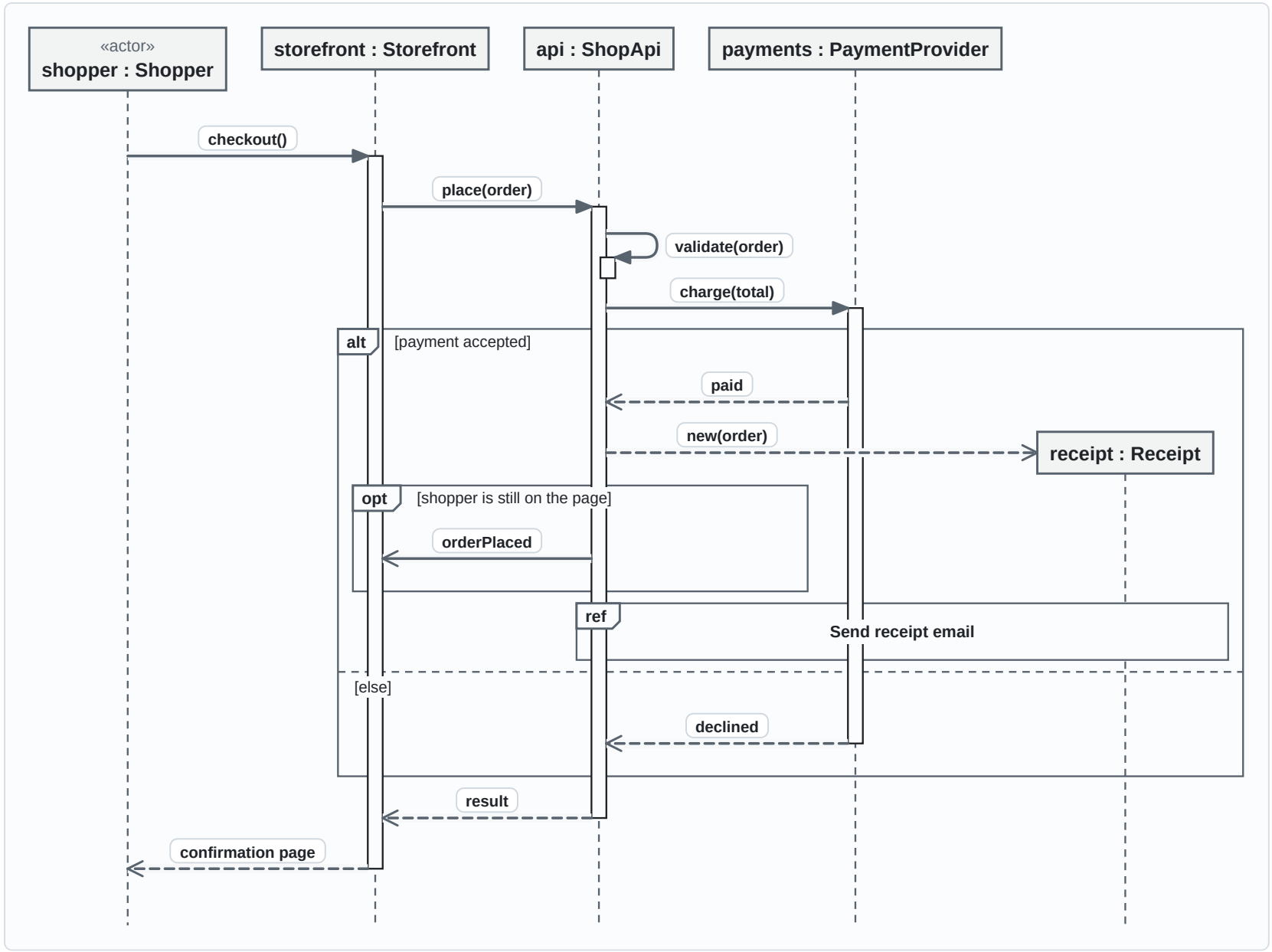
Legend **Call** the sender waits for it **Reply** the answer to a call **Duration constraint** how long may pass between two changes **Time constraint** when a change may happen **States** the lifeline steps between its states

Elements 2

Card	Kind	Members
worker : OrderWorker	Lifeline	0 ms: Idle 100 ms: Waiting for payment 760 ms: Recording payment 900 ms: Idle charge(total) to payments at 100 ms → 140 ms Duration constraint {<= 800 ms} on worker from 100 ms to 760 ms Time constraint {<= 1000 ms} on worker at 900 ms
payments : PaymentProvider	Lifeline	0 ms: Idle 140 ms: Authorising 480 ms: Capturing 720 ms: Idle paid to worker at 720 ms → 760 ms

Lines 2

From → to	Label	Kind
worker : OrderWorker → payments : PaymentProvider	charge(total)	Message
payments : PaymentProvider → worker : OrderWorker	paid	Message



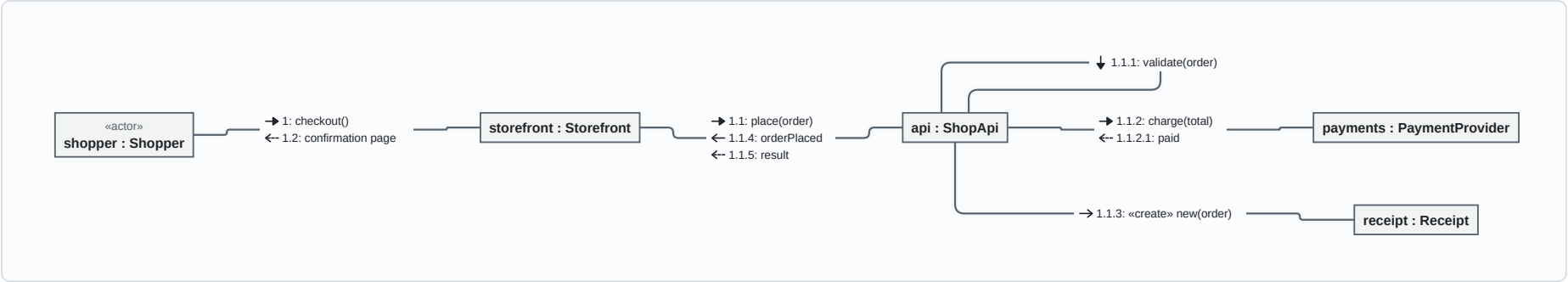
Legend → **Call** the sender waits for it —> **Async** the sender goes on --> **Reply** the answer to a call --> **Create** creates the lifeline it points to

Elements 5

Card	Kind	Members
shopper : Shopper	Lifeline	
storefront : Storefront	Lifeline	
api : ShopApi	Lifeline	
payments : PaymentProvider	Lifeline	
receipt : Receipt	Lifeline	

Lines 10

From → to	Label	Kind
shopper : Shopper → storefront : Storefront	checkout()	Message
storefront : Storefront → api : ShopApi	place(order)	Message
api : ShopApi → api : ShopApi	validate(order)	Message
api : ShopApi → payments : PaymentProvider	charge(total)	Message
payments : PaymentProvider → api : ShopApi	paid	Message
api : ShopApi → receipt : Receipt	new(order)	Message
api : ShopApi → storefront : Storefront	orderPlaced	Message
payments : PaymentProvider → api : ShopApi	declined	Message
api : ShopApi → storefront : Storefront	result	Message
storefront : Storefront → shopper : Shopper	confirmation page	Message



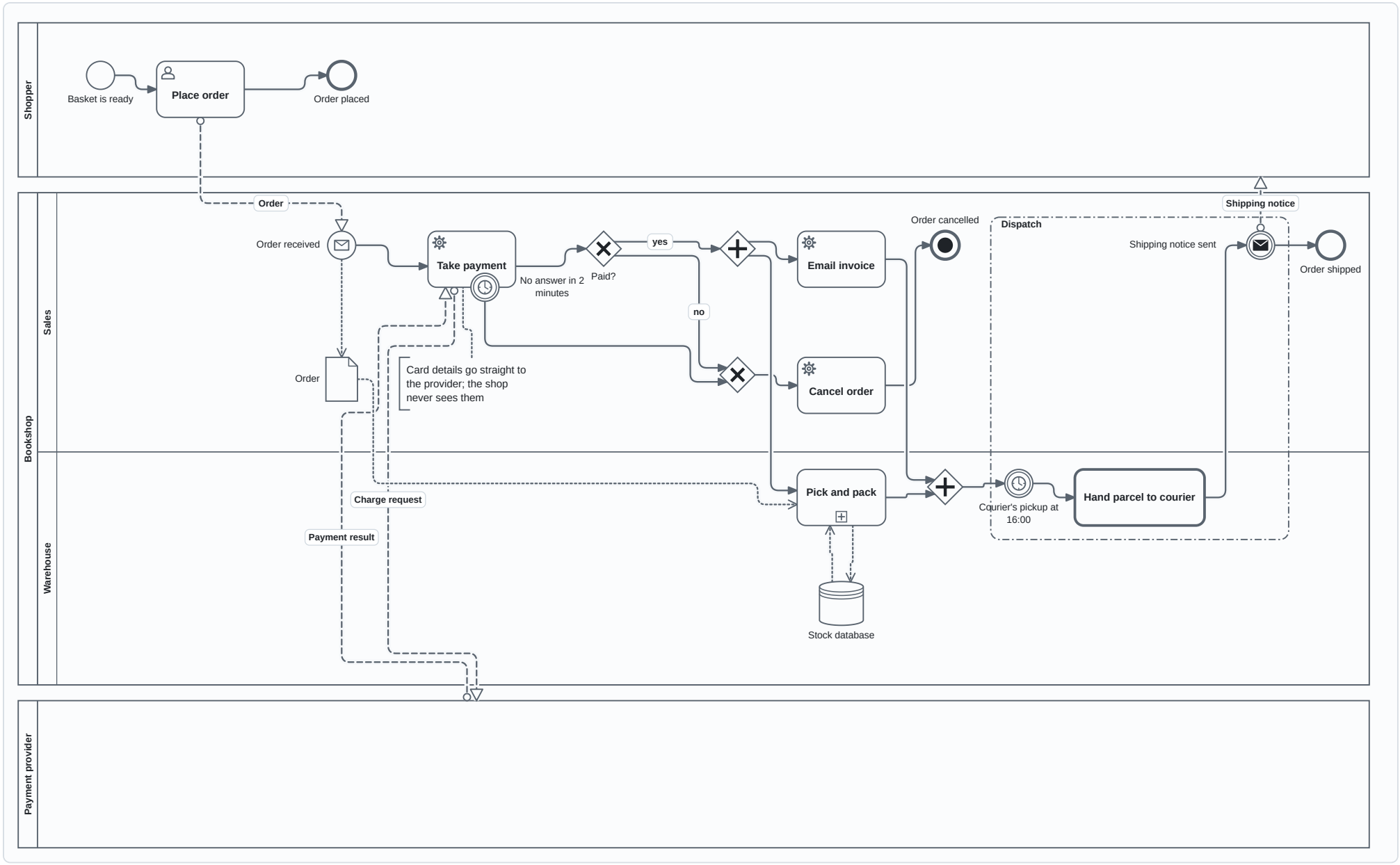
Legend — **Connector** a link messages are sent along → **Call** waits for its reply ⇢ **Async** sent without waiting ⇢ **Reply** the answer to a call → **Create** «create»: makes its receiver

Elements 5

Card	Kind	Members
shopper : Shopper	Lifeline	
storefront : Storefront	Lifeline	
api : ShopApi	Lifeline	
payments : PaymentProvider	Lifeline	
receipt : Receipt	Lifeline	

Lines 14

From → to	Label	Kind
shopper : Shopper → storefront : Storefront		Connector
storefront : Storefront → api : ShopApi		Connector
api : ShopApi → api : ShopApi		Connector
api : ShopApi → payments : PaymentProvider		Connector
api : ShopApi → receipt : Receipt		Connector
shopper : Shopper → storefront : Storefront	checkout()	Message
storefront : Storefront → api : ShopApi	place(order)	Message
api : ShopApi → api : ShopApi	validate(order)	Message
api : ShopApi → payments : PaymentProvider	charge(total)	Message
payments : PaymentProvider → api : ShopApi	paid	Message
api : ShopApi → receipt : Receipt	new(order)	Message
api : ShopApi → storefront : Storefront	orderPlaced	Message
api : ShopApi → storefront : Storefront	result	Message
storefront : Storefront → shopper : Shopper	confirmation page	Message



Legend → **Sequence flow** the order of steps in one pool ⇢ **Message flow** a message between pools ····· **Association** a note or link to an element ····> **Data association** data read or written

Elements 25

Card	Kind	Members
Shopper	Pool	
Basket is ready	Start event	in Shopper
Place order	User task	in Shopper
Order placed	End event	in Shopper
Bookshop	Pool	Sales Warehouse
Order received	Message start event	in Sales
Take payment	Service task	in Sales
No answer in 2 minutes	Timer boundary event	on Take payment
Paid?	Exclusive gateway	in Sales
Exclusive gateway	Exclusive gateway	in Sales
Cancel order	Service task	in Sales Nothing to refund: no payment was taken.
Order cancelled	Terminate end event	in Sales
Parallel gateway	Parallel gateway	in Sales
Email invoice	Service task	in Sales Sent to the address the order gives.
Pick and pack	Sub-process (collapsed)	in Warehouse
Parallel gateway	Parallel gateway	in Warehouse
Courier's pickup at 16:00	Timer intermediate catch event	in Warehouse
Hand parcel to courier	Call activity	in Warehouse
Shipping notice sent	Message intermediate throw event	in Sales
Order shipped	End event	in Sales
Payment provider	Pool	

Elements 25 (continued)

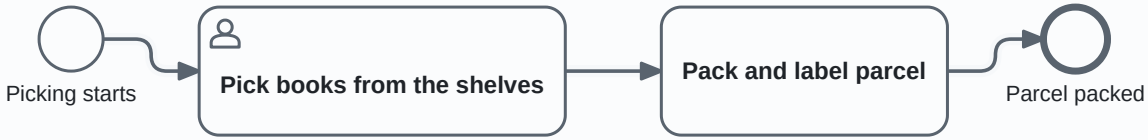
Card	Kind	Members
Order	Data object	in Sales
Stock database	Data store	in Warehouse
Card details go straight to the provider; the shop never sees them	Text annotation	in Sales
Dispatch	Group	

Lines 26

From → to	Label	Kind
Basket is ready → Place order		Sequence flow
Place order → Order placed		Sequence flow
Order received → Take payment		Sequence flow
Take payment → Paid?		Sequence flow
Paid? → Parallel gateway	yes	Sequence flow
Paid? → Exclusive gateway	no	Sequence flow
No answer in 2 minutes → Exclusive gateway		Sequence flow
Exclusive gateway → Cancel order		Sequence flow
Cancel order → Order cancelled		Sequence flow
Parallel gateway → Pick and pack		Sequence flow
Parallel gateway → Email invoice		Sequence flow
Pick and pack → Parallel gateway		Sequence flow
Email invoice → Parallel gateway		Sequence flow
Parallel gateway → Courier's pickup at 16:00		Sequence flow
Courier's pickup at 16:00 → Hand parcel to courier		Sequence flow
Hand parcel to courier → Shipping notice sent		Sequence flow
Shipping notice sent → Order shipped		Sequence flow

Lines 26 (continued)

From → to	Label	Kind
— Place order → Order received	Order	Message flow
— Take payment → Payment provider	Charge request	Message flow
— Payment provider → Take payment	Payment result	Message flow
— Shipping notice sent → Shopper	Shipping notice	Message flow
— Order received → Order		Data association
— Order → Pick and pack		Data association
— Stock database → Pick and pack		Data association
— Pick and pack → Stock database		Data association
— Card details go straight to the provider; the shop never sees them → Take payment		Association



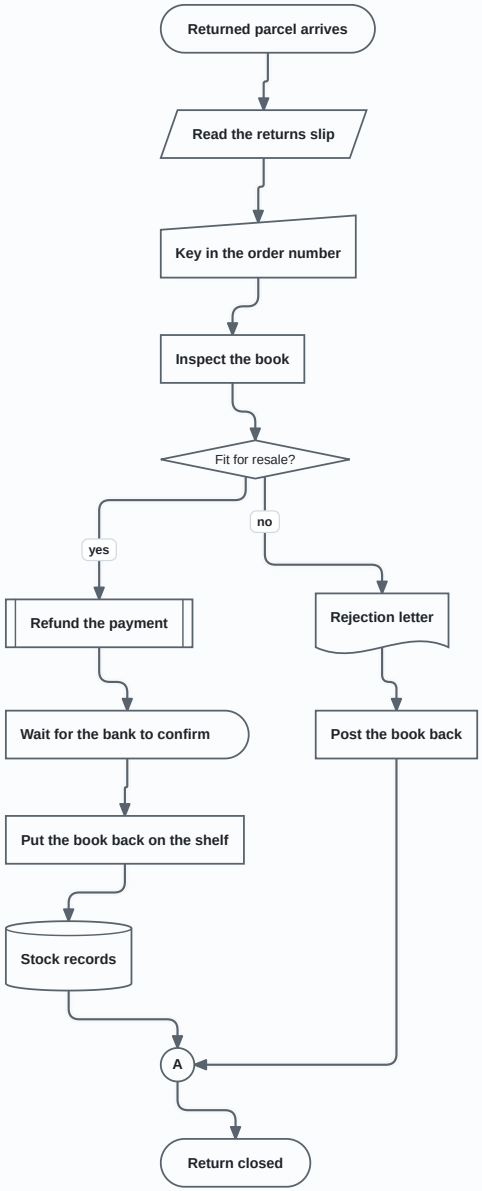
Legend → **Sequence flow** the order of steps in one pool

Elements 4

Card	Kind	Members
Picking starts	Start event	
Pick books from the shelves	User task	
Pack and label parcel	Task	
Parcel packed	End event	

Lines 3

From → to	Label	Kind
Picking starts → Pick books from the shelves		Sequence flow
Pick books from the shelves → Pack and label parcel		Sequence flow
Pack and label parcel → Parcel packed		Sequence flow



Legend **Flow** the next step · **Terminator** a start or an end · **Process** a step · **Decision** a question, an exit per answer · **Input/output** data read or written · **Predefined process** steps written out elsewhere · **Document** a document made or read · **Database** stored data · **Manual input** data keyed in by hand · **Delay** a wait · **Connector** flows joined, or carried on elsewhere

Elements 13

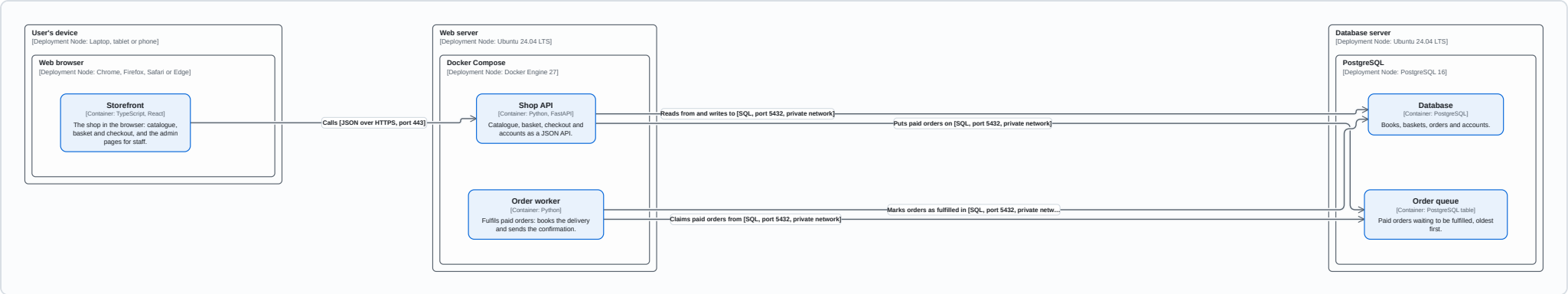
Card	Kind	Members
Returned parcel arrives	Terminator	
Read the returns slip	Input/output	
Key in the order number	Manual input	
Inspect the book	Process	
Fit for resale?	Decision	
Refund the payment	Predefined process	
Wait for the bank to confirm	Delay	
Put the book back on the shelf	Process	
Stock records	Database	
Rejection letter	Document	
Post the book back	Process	
A	Connector	
Return closed	Terminator	

Lines 13

From → to	Label	Kind
Returned parcel arrives → Read the returns slip		Flow
Read the returns slip → Key in the order number		Flow
Key in the order number → Inspect the book		Flow
Inspect the book → Fit for resale?		Flow
Fit for resale? → Refund the payment	yes	Flow
Fit for resale? → Rejection letter	no	Flow
Refund the payment → Wait for the bank to confirm		Flow

Lines 13 (continued)

From → to	Label	Kind
Wait for the bank to confirm → Put the book back on the shelf		Flow
Put the book back on the shelf → Stock records		Flow
Stock records → A		Flow
Rejection letter → Post the book back		Flow
Post the book back → A		Flow
A → Return closed		Flow



Legend **Deployment node** a host, runtime or device; it holds what runs on it **Container instance** a container deployed here, drawn as its container **Relationship** uses, with its technology in brackets

Elements 11

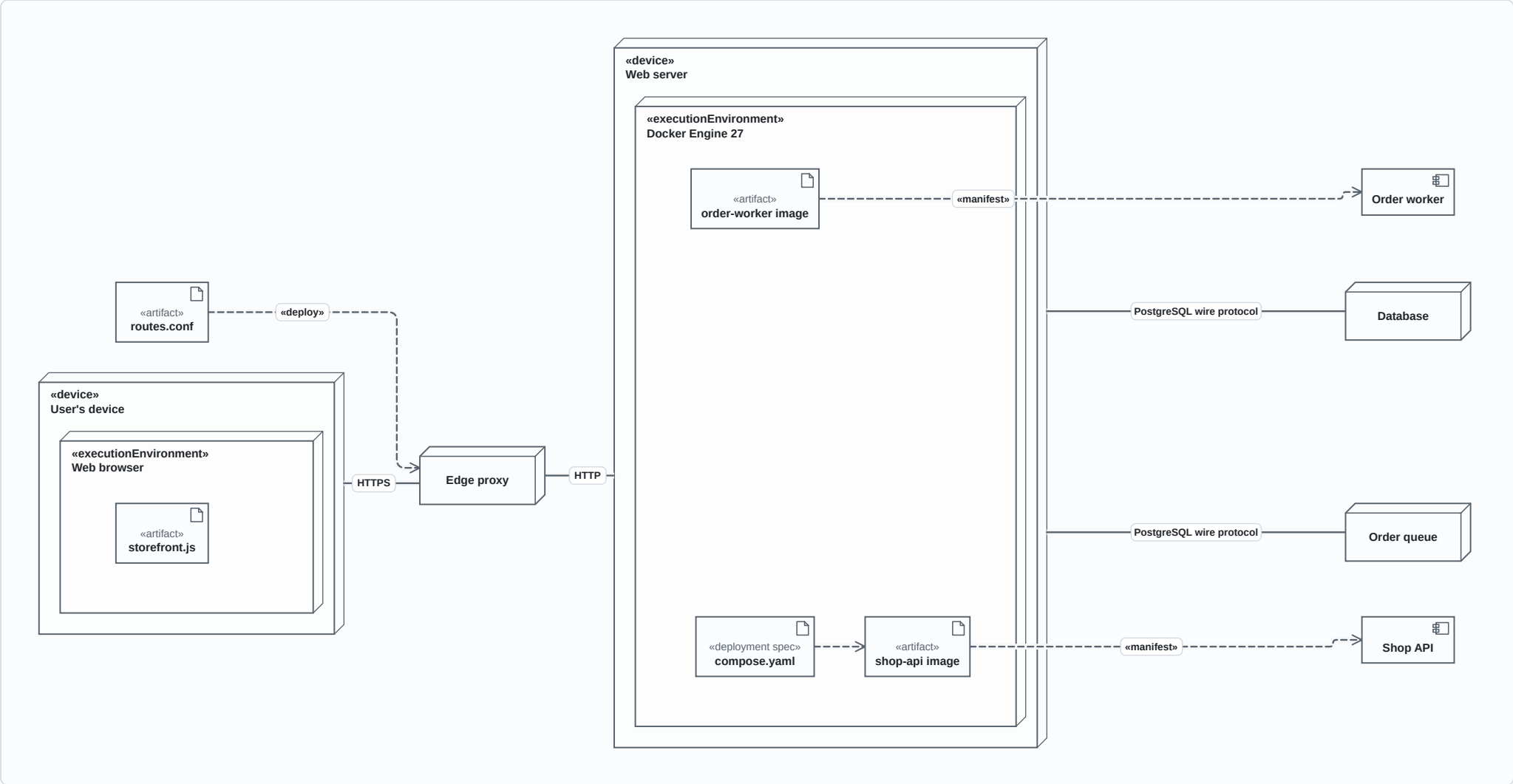
Card	Kind	Members
User's device	Deployment node	Laptop, tablet or phone A shopper's or a staff member's own device.
Web browser	Deployment node	Chrome, Firefox, Safari or Edge
Storefront	Container instance	Instance of Storefront TypeScript, React The shop in the browser: catalogue, basket and checkout, and the admin pages for staff.
Web server	Deployment node	Ubuntu 24.04 LTS The web tier's server.
Docker Compose	Deployment node	Docker Engine 27
Shop API	Container instance	Instance of Shop API Python, FastAPI Catalogue, basket, checkout and accounts as a JSON API.
Order worker	Container instance	Instance of Order worker Python Fulfils paid orders: books the delivery and sends the confirmation.
Database server	Deployment node	Ubuntu 24.04 LTS Backed up every night.
PostgreSQL	Deployment node	PostgreSQL 16

Elements 11 (continued)

Card	Kind	Members
Database	Container instance	Instance of Database PostgreSQL Books, baskets, orders and accounts.
Order queue	Container instance	Instance of Order queue PostgreSQL table Paid orders waiting to be fulfilled, oldest first.

Lines 5

From → to	Label	Kind
Storefront → Shop API	Calls [JSON over HTTPS, port 443]	Relationship
Shop API → Database	Reads from and writes to [SQL, port 5432, private network]	Relationship
Shop API → Order queue	Puts paid orders on [SQL, port 5432, private network]	Relationship
Order worker → Order queue	Claims paid orders from [SQL, port 5432, private network]	Relationship
Order worker → Database	Marks orders as fulfilled in [SQL, port 5432, private network]	Relationship



Legend --> **Deployment** «deploy»: the artifact runs on the target --> **Manifestation** «manifest»: the artifact is built from the component — **Communication path** the two exchange messages

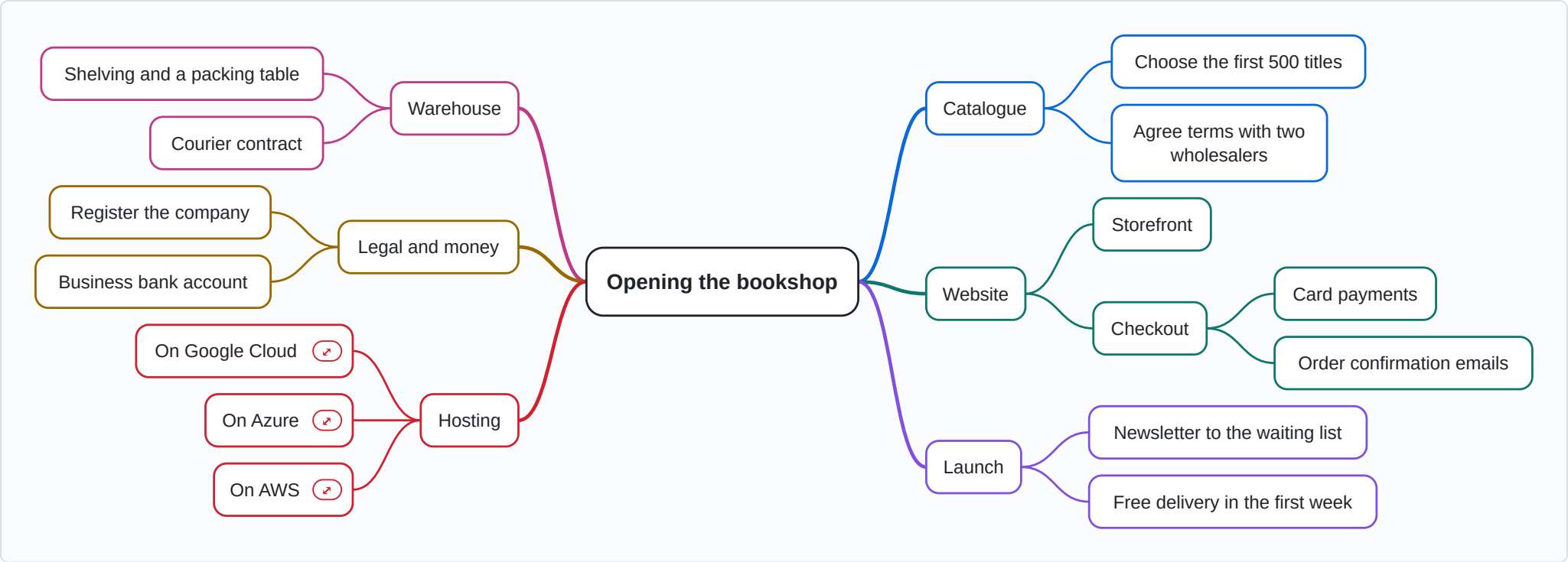
--> **Dependency** uses the one it points to

Elements 14

Card	Kind	Members
User's device	Device	
Web browser	Execution environment	storefront.js
storefront.js	Artifact	
Edge proxy	Node	routes.conf, by «deploy»
routes.conf	Artifact	
Web server	Device	
Docker Engine 27	Execution environment	shop-api image order-worker image compose.yaml
shop-api image	Artifact	manifests Shop API
order-worker image	Artifact	manifests Order worker
compose.yaml	Deployment specification	
Database	Node	
Order queue	Node	
Shop API	Component	
Order worker	Component	

Lines 8

From → to	Label	Kind
routes.conf → Edge proxy		Deployment
shop-api image → Shop API		Manifestation
order-worker image → Order worker		Manifestation
compose.yaml → shop-api image		Dependency
User's device → Edge proxy	HTTPS	Communication path
Edge proxy → Web server	HTTP	Communication path
Web server → Database	PostgreSQL wire protocol	Communication path
Web server → Order queue	PostgreSQL wire protocol	Communication path



Legend **File / Doc** a document or config **⤢ Stacked** opens a view of its own

Elements 22

Card	Kind	Members
Opening the bookshop	Topic	Depth 0 The root
Catalogue	Topic	Depth 1 Under Opening the bookshop
Choose the first 500 titles	Topic	Depth 2 Under Catalogue
Agree terms with two wholesalers	Topic	Depth 2 Under Catalogue
Website	Topic	Depth 1 Under Opening the bookshop

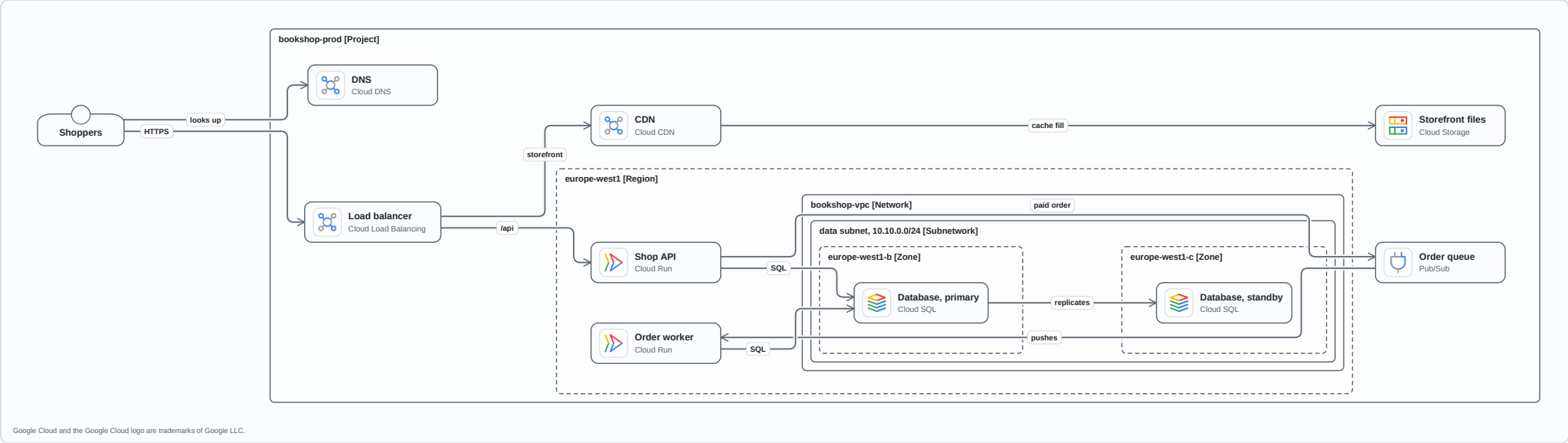
Elements 22 (continued)

Card	Kind	Members
Storefront	Topic	Depth 2 Under Website
Checkout	Topic	Depth 2 Under Website
Card payments	Topic	Depth 3 Under Checkout
Order confirmation emails	Topic	Depth 3 Under Checkout
Launch	Topic	Depth 1 Under Opening the bookshop

Elements 22 (continued)

Card	Kind	Members
Newsletter to the waiting list	Topic	Depth 2 Under Launch
Free delivery in the first week	Topic	Depth 2 Under Launch
Warehouse	Topic	Depth 1 Under Opening the bookshop
Shelving and a packing table	Topic	Depth 2 Under Warehouse
Courier contract	Topic	Depth 2 Under Warehouse
Legal and money	Topic	Depth 1 Under Opening the bookshop
Register the company	Topic	Depth 2 Under Legal and money
Business bank account	Topic	Depth 2 Under Legal and money
Hosting	Topic	Depth 1 Under Opening the bookshop
On Google Cloud ⤢	Topic	Depth 2 Under Hosting
On Azure ⤢	Topic	Depth 2 Under Hosting
On AWS ⤢	Topic	Depth 2 Under Hosting

Icons from a pack you fetch: umlimator.py pack fetch gcp, after Google's terms; glyphs and full names without it. Umlimator is not affiliated with, endorsed by or sponsored by Amazon Web Services, Google or Microsoft.



Legend **Cloud DNS** icon Cloud-DNS **Cloud Load Balancing** icon Cloud-Load-Balancing **Cloud CDN** icon Cloud-CDN **Cloud Storage** icon Cloud-Storage
Pub/Sub icon Pub-Sub **Cloud Run** icon Cloud-Run **Cloud SQL** icon Cloud-SQL **Project** a group, its kind in brackets **Region** a group, its kind in brackets
Network a group, its kind in brackets **Subnetwork** a group, its kind in brackets **Zone** a group, its kind in brackets **User** a person who uses the system
Path a request or a flow of data, to the arrow Google Cloud and the Google Cloud logo are trademarks of Google LLC.

Elements 16

Card	Kind	Members
Shoppers	User	
bookshop-prod	Project	
DNS	Product	Cloud DNS icon Cloud-DNS in bookshop-prod [Project]
Load balancer	Product	Cloud Load Balancing icon Cloud-Load-Balancing in bookshop-prod [Project]

Elements 16 (continued)

Card	Kind	Members
CDN	Product	Cloud CDN icon Cloud-CDN in bookshop-prod [Project]
Storefront files	Product	Cloud Storage icon Cloud-Storage in bookshop-prod [Project]
Order queue	Product	Pub/Sub icon Pub-Sub in bookshop-prod [Project]
europe-west1	Region	

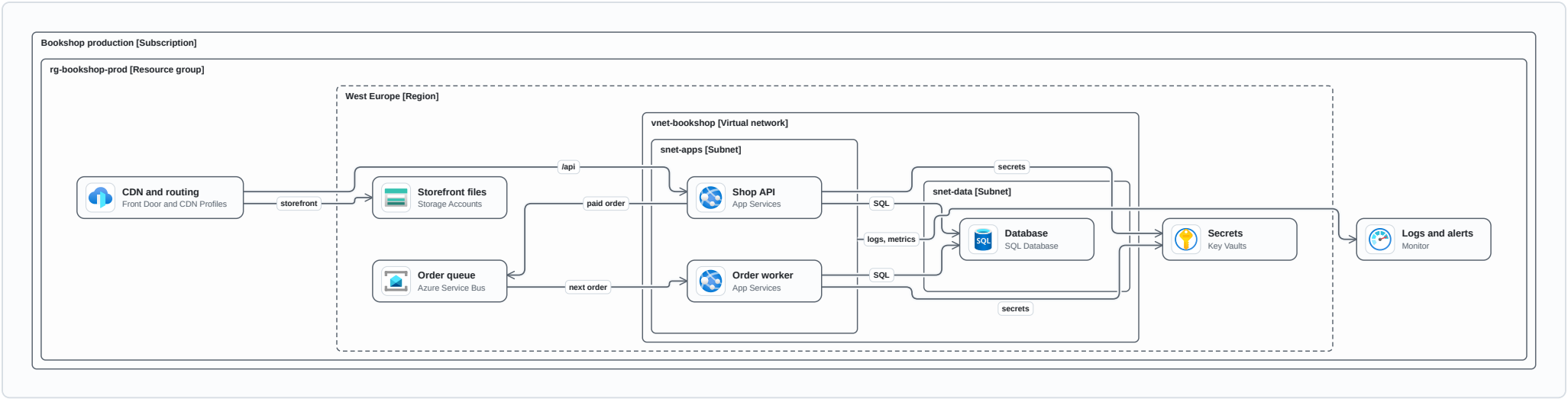
Elements 16 (continued)

Card	Kind	Members
Shop API	Product	Cloud Run icon Cloud-Run in europe-west1 [Region]
Order worker	Product	Cloud Run icon Cloud-Run in europe-west1 [Region]
bookshop-vpc	Network	
data subnet, 10.10.0.0/24	Subnetwork	
europe-west1-b	Zone	
Database, primary	Product	Cloud SQL icon Cloud-SQL in europe-west1-b [Zone]
europe-west1-c	Zone	
Database, standby	Product	Cloud SQL icon Cloud-SQL in europe-west1-c [Zone]

Lines 10

From → to	Label	Kind
Shoppers → DNS	looks up	Path
Shoppers → Load balancer	HTTPS	Path
Load balancer → CDN	storefront	Path
CDN → Storefront files	cache fill	Path
Load balancer → Shop API	/api	Path
Shop API → Database, primary	SQL	Path
Shop API → Order queue	paid order	Path
Order queue → Order worker	pushes	Path
Order worker → Database, primary	SQL	Path
Database, primary → Database, standby	replicates	Path

Icons from a pack you fetch: umlimator.py pack fetch azure, after Microsoft's terms; glyphs and full names without it. Umlimator is not affiliated with, endorsed by or sponsored by Amazon Web Services, Google or Microsoft.



Legend **Front Door and CDN Profiles** icon Front-Door-and-CDN-Profiles · **Monitor** icon Monitor · **Storage Accounts** icon Storage-Accounts · **Azure Service Bus** icon Service-Bus · **Key Vaults** icon Key-Vaults · **App Services** icon App-Services · **SQL Database** icon SQL-Database · **Subscription** a group, its kind in brackets · **Resource group** a group, its kind in brackets · **Region** a group, its kind in brackets · **Virtual network** a group, its kind in brackets · **Subnet** a group, its kind in brackets · **Connector** a request or a flow of data, to the arrow

Elements 14

Card	Kind	Members
Bookshop production	Subscription	
rg-bookshop-prod	Resource group	
CDN and routing	Service	Front Door and CDN Profiles icon Front-Door-and-CDN-Profiles in rg-bookshop-prod [Resource group]
Logs and alerts	Service	Monitor icon Monitor in rg-bookshop-prod [Resource group]
West Europe	Region	
Storefront files	Service	Storage Accounts icon Storage-Accounts in West Europe [Region]

Elements 14 (continued)

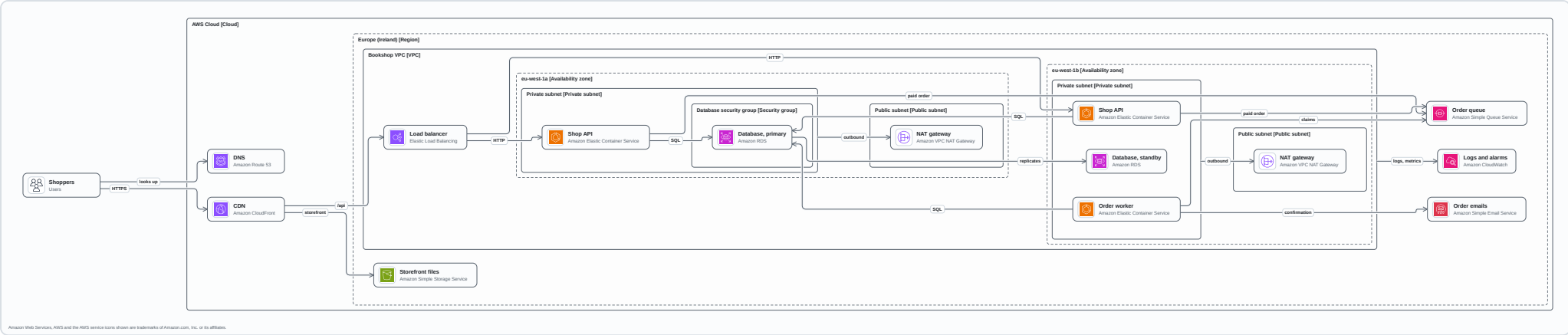
Card	Kind	Members
Order queue	Service	Azure Service Bus icon Service-Bus in West Europe [Region]
Secrets	Service	Key Vaults icon Key-Vaults in West Europe [Region]
vnet-bookshop	Virtual network	
snet-apps	Subnet	
Shop API	Service	App Services icon App-Services in snet-apps [Subnet]

Elements 14 (continued)

Card	Kind	Members
Order worker	Service	App Services icon App-Services in snet-apps [Subnet]
snet-data	Subnet	
Database	Service	SQL Database icon SQL-Database in snet-data [Subnet]

Lines 9

From → to	Label	Kind
CDN and routing → Storefront files	storefront	Connector
CDN and routing → Shop API	/api	Connector
Shop API → Database	SQL	Connector
Shop API → Order queue	paid order	Connector
Order queue → Order worker	next order	Connector
Order worker → Database	SQL	Connector
Shop API → Secrets	secrets	Connector
Order worker → Secrets	secrets	Connector
snet-apps → Logs and alerts	logs, metrics	Connector



Legend **Users** icon Users **Amazon Route 53** icon Amazon-Route-53 **Amazon CloudFront** icon Amazon-CloudFront
Amazon Simple Storage Service icon Amazon-Simple-Storage-Service **Amazon Simple Queue Service** icon Amazon-Simple-Queue-Service
Amazon Simple Email Service icon Amazon-Simple-Email-Service **Amazon CloudWatch** icon Amazon-CloudWatch **Elastic Load Balancing** icon Elastic-Load-Balancing
Amazon VPC NAT Gateway icon Amazon-VPC_NAT-Gateway **Amazon Elastic Container Service** icon Amazon-Elastic-Container-Service **Amazon RDS** icon Amazon-RDS
Cloud a group, its kind in brackets **Region** a group, its kind in brackets **VPC** a group, its kind in brackets **Availability zone** a group, its kind in brackets
Public subnet a group, its kind in brackets **Private subnet** a group, its kind in brackets **Security group** a group, its kind in brackets **Connection** a request or a flow of data, to the arrow

Amazon Web Services, AWS and the AWS service icons shown are trademarks of Amazon.com, Inc. or its affiliates.

Elements 25

Card	Kind	Members
Shoppers	Resource	Users icon Users
AWS Cloud	Cloud	
DNS	Service	Amazon Route 53 icon Amazon-Route-53 in AWS Cloud [Cloud]
CDN	Service	Amazon CloudFront icon Amazon-CloudFront in AWS Cloud [Cloud]
Europe (Ireland)	Region	
Storefront files	Service	Amazon Simple Storage Service icon Amazon-Simple-Storage-Service in Europe (Ireland) [Region]

Elements 25 (continued)

Card	Kind	Members
Order queue	Service	Amazon Simple Queue Service icon Amazon-Simple-Queue-Service in Europe (Ireland) [Region]
Order emails	Service	Amazon Simple Email Service icon Amazon-Simple-Email-Service in Europe (Ireland) [Region]
Logs and alarms	Service	Amazon CloudWatch icon Amazon-CloudWatch in Europe (Ireland) [Region]
Bookshop VPC	VPC	
Load balancer	Service	Elastic Load Balancing icon Elastic-Load-Balancing in Bookshop VPC [VPC]
eu-west-1a	Availability zone	

Elements 25 (continued)

Card	Kind	Members
Public subnet	Public subnet	
NAT gateway	Resource	Amazon VPC NAT Gateway icon Amazon-VPC_NAT-Gateway in Public subnet [Public subnet]
Private subnet	Private subnet	
Shop API	Service	Amazon Elastic Container Service icon Amazon-Elastic-Container-Service in Private subnet [Private subnet]
Database security group	Security group	
Database, primary	Service	Amazon RDS icon Amazon-RDS in Database security group [Security group]
eu-west-1b	Availability zone	
Public subnet	Public subnet	
NAT gateway	Resource	Amazon VPC NAT Gateway icon Amazon-VPC_NAT-Gateway in Public subnet [Public subnet]
Private subnet	Private subnet	
Shop API	Service	Amazon Elastic Container Service icon Amazon-Elastic-Container-Service in Private subnet [Private subnet]
Order worker	Service	Amazon Elastic Container Service icon Amazon-Elastic-Container-Service in Private subnet [Private subnet]
Database, standby	Service	Amazon RDS icon Amazon-RDS in Private subnet [Private subnet]

Lines 17

From → to	Label	Kind
Shoppers → DNS	looks up	Connection
Shoppers → CDN	HTTPS	Connection
CDN → Storefront files	storefront	Connection
CDN → Load balancer	/api	Connection
Load balancer → Shop API	HTTP	Connection
Load balancer → Shop API	HTTP	Connection
Shop API → Database, primary	SQL	Connection
Shop API → Database, primary	SQL	Connection
Database, primary → Database, standby	replicates	Connection
Shop API → Order queue	paid order	Connection
Shop API → Order queue	paid order	Connection
Order worker → Order queue	claims	Connection
Order worker → Database, primary	SQL	Connection
Order worker → Order emails	confirmation	Connection
Private subnet → NAT gateway	outbound	Connection
Private subnet → NAT gateway	outbound	Connection
Bookshop VPC → Logs and alarms	logs, metrics	Connection