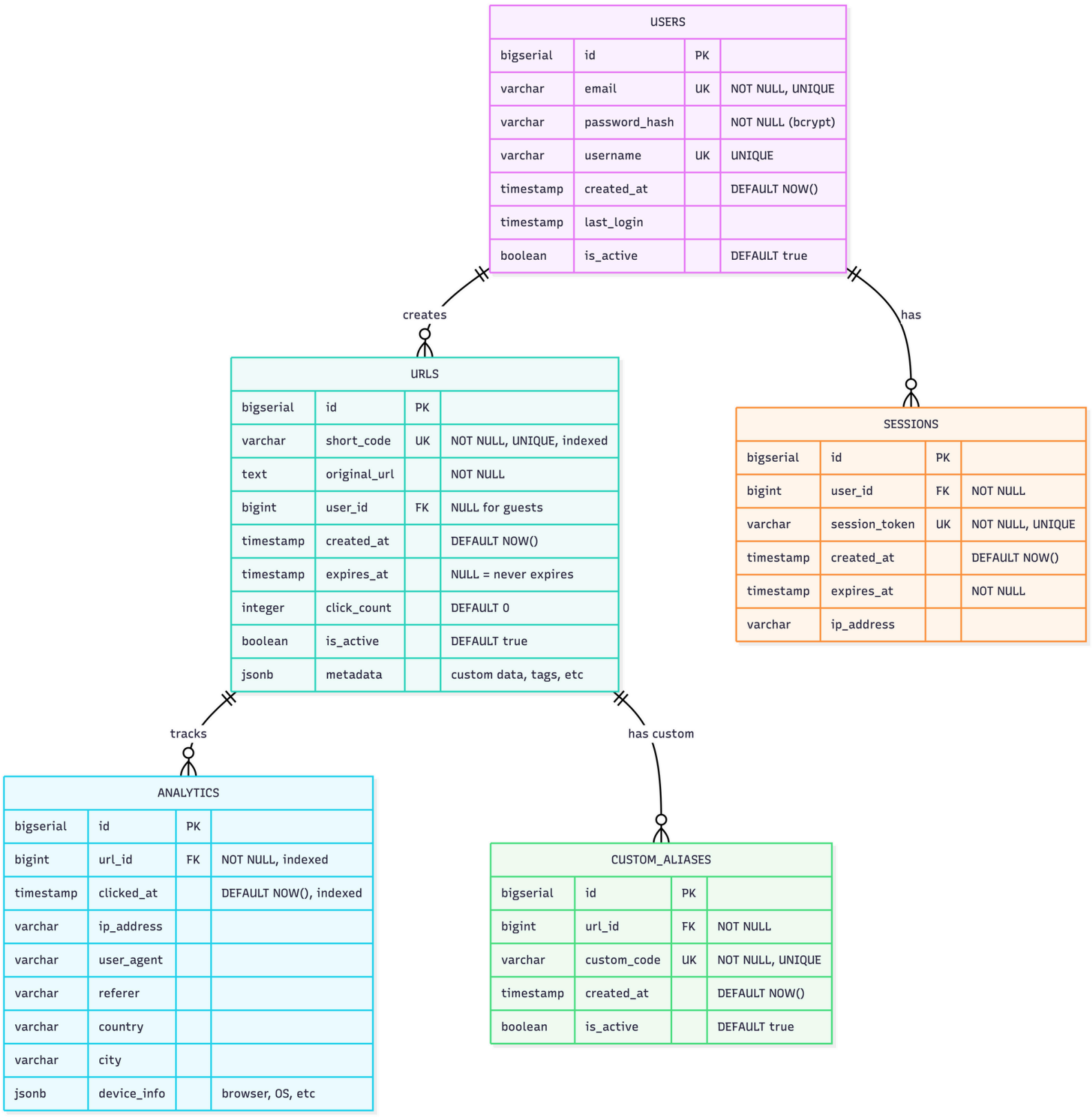
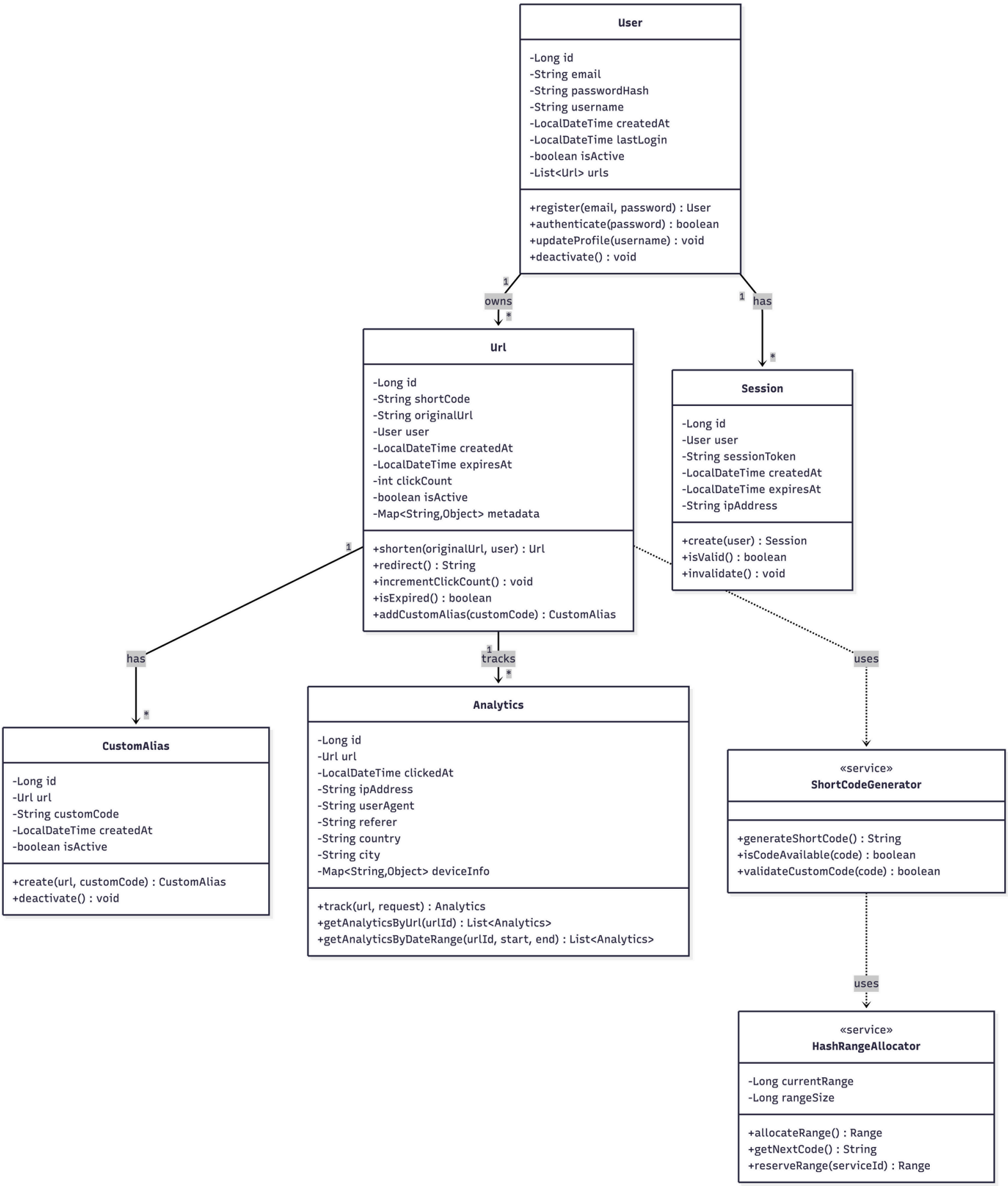
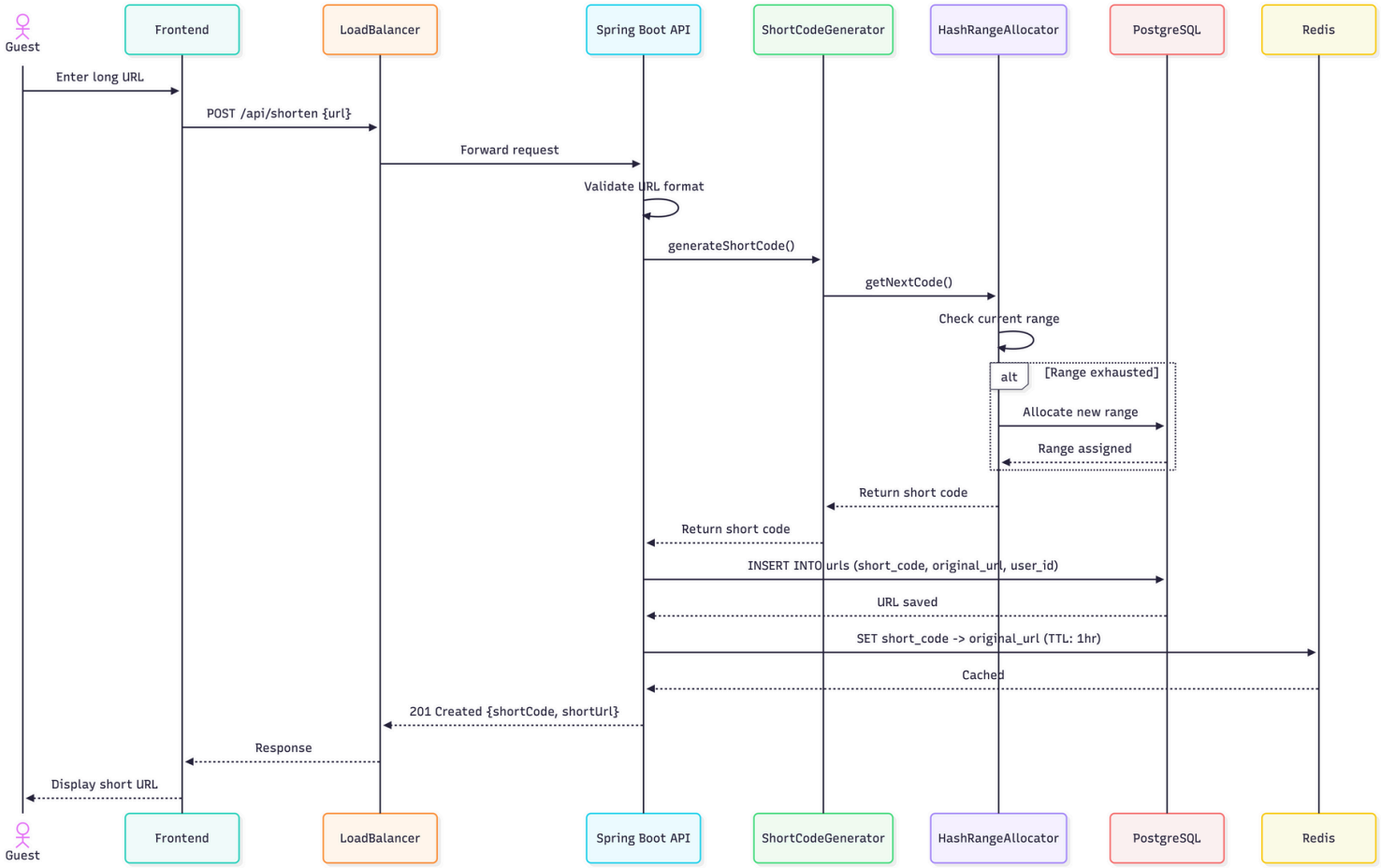
URL SHORTENER – DATABASE SCHEMA (ER DIAGRAM)



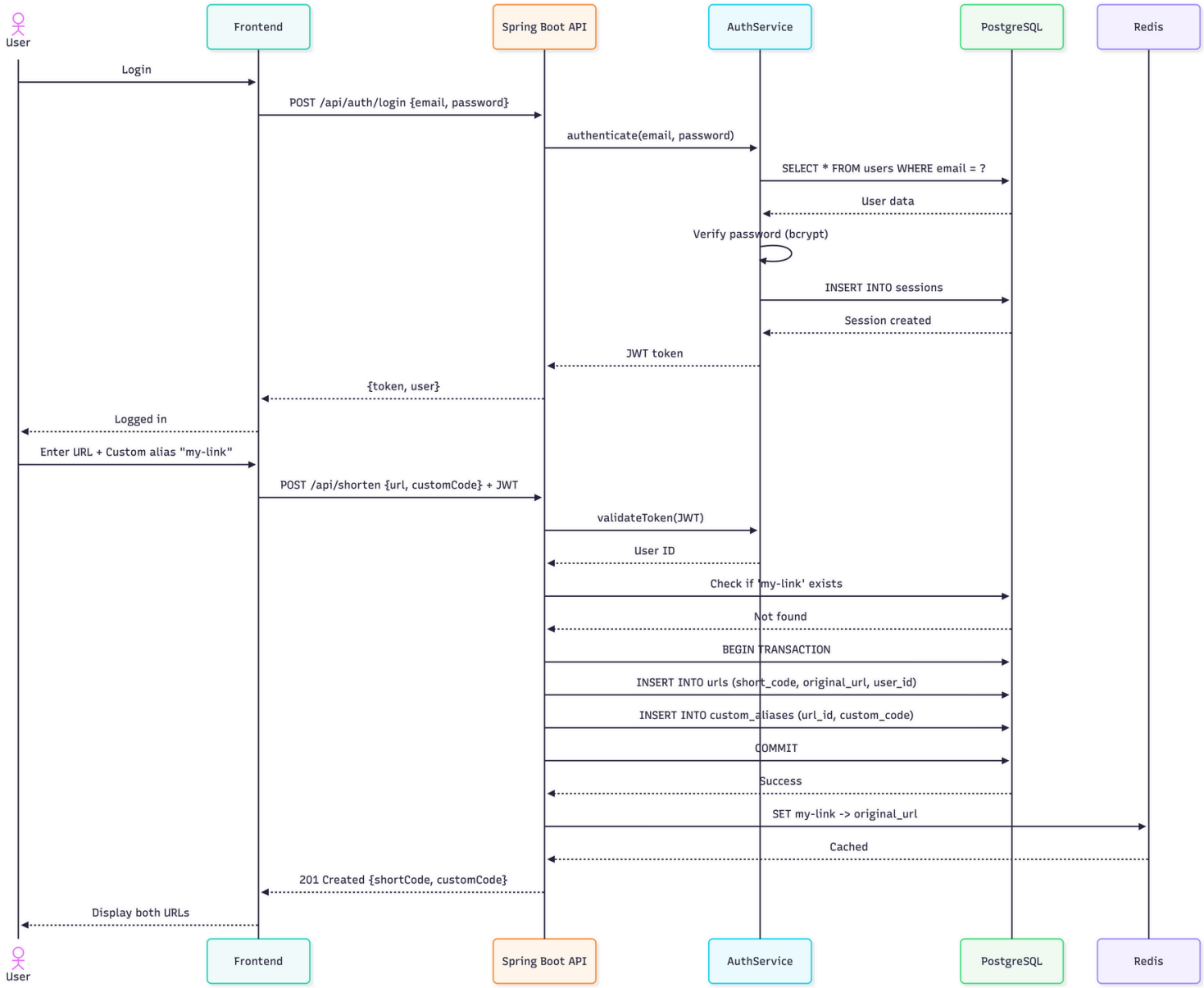
CLASS DIAGRAM – BACKEND DOMAIN MODEL



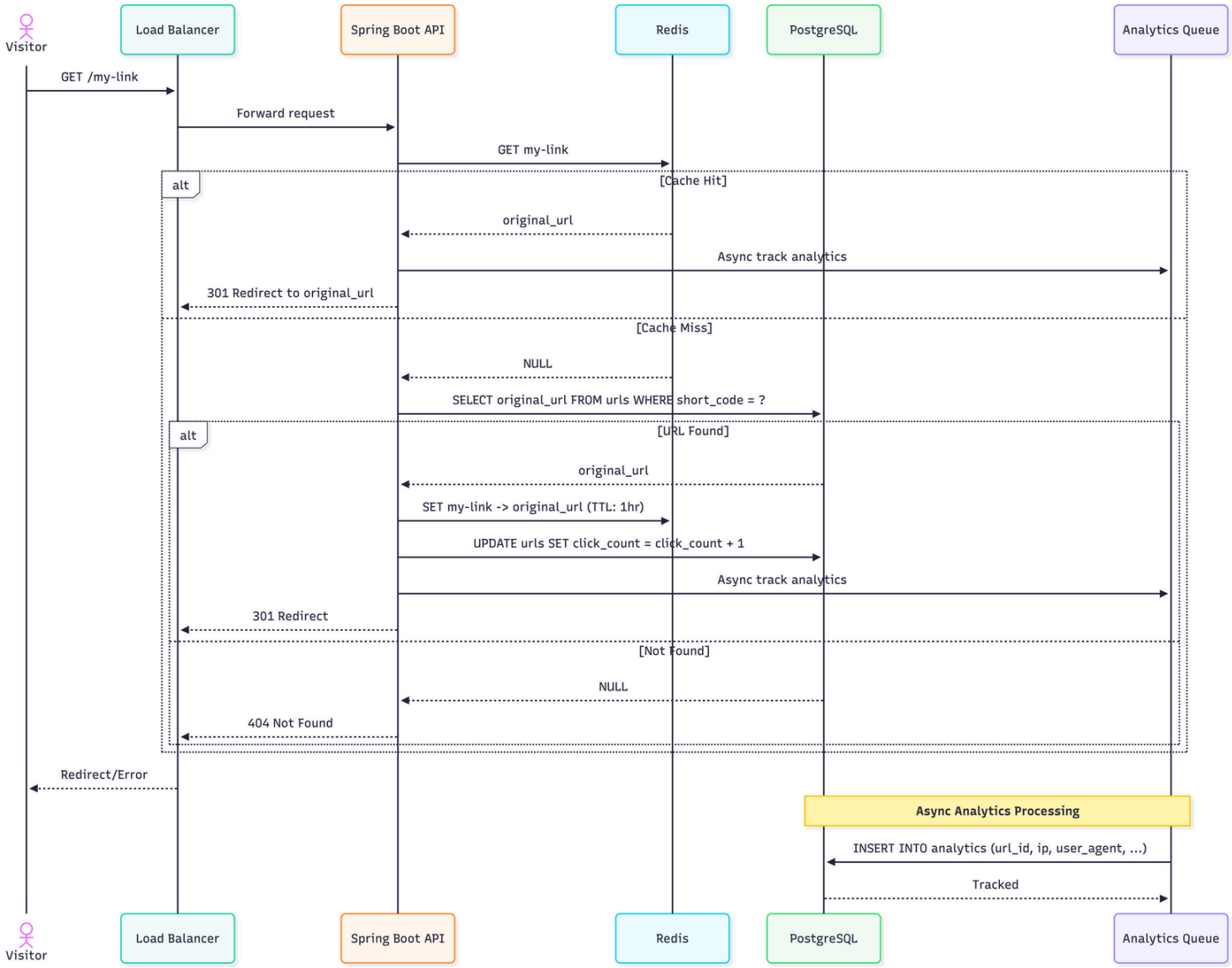
SEQUENCE DIAGRAM – GUEST SHORTENING URL



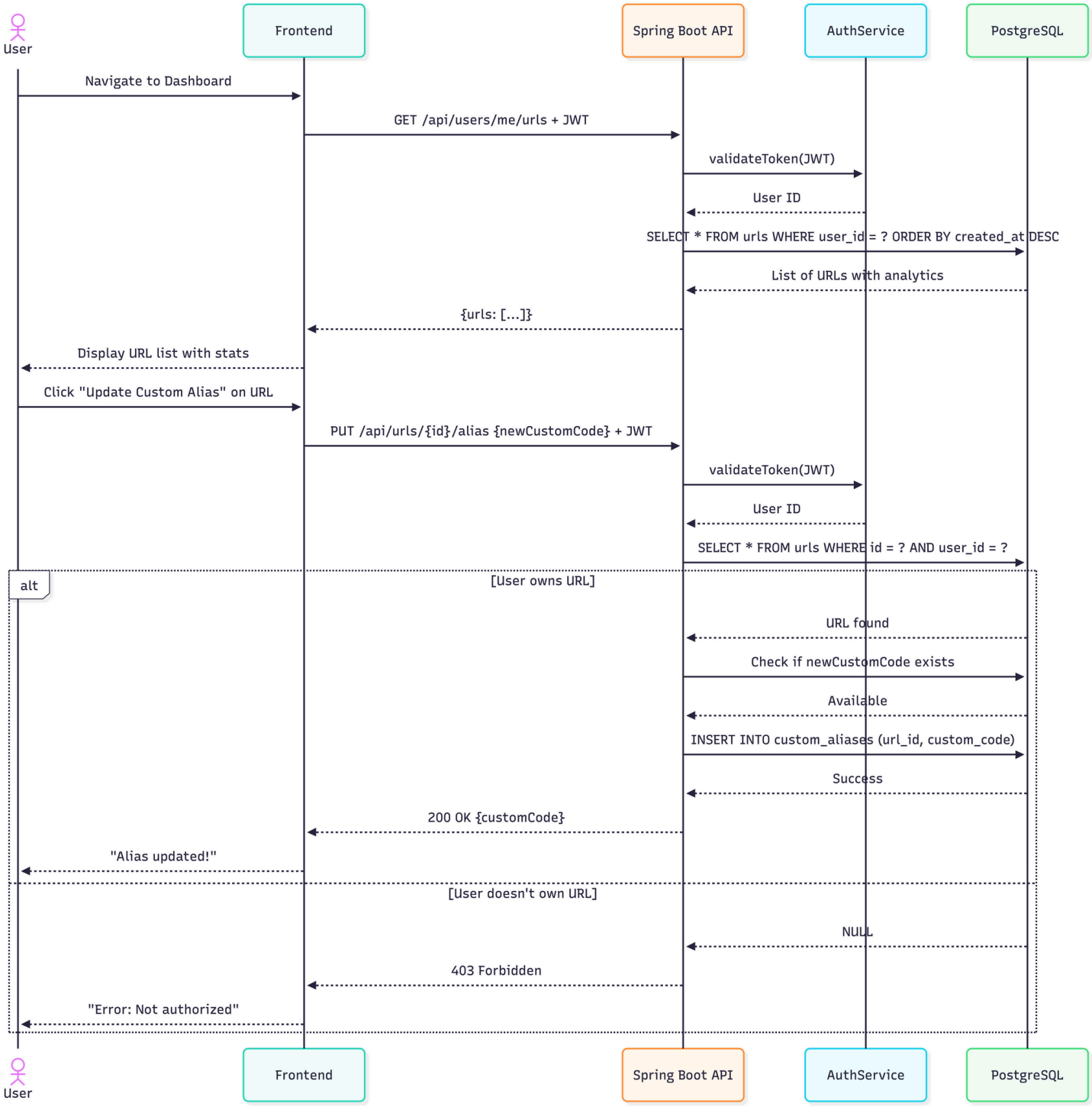
SEQUENCE DIAGRAM – USER WITH CUSTOM ALIAS



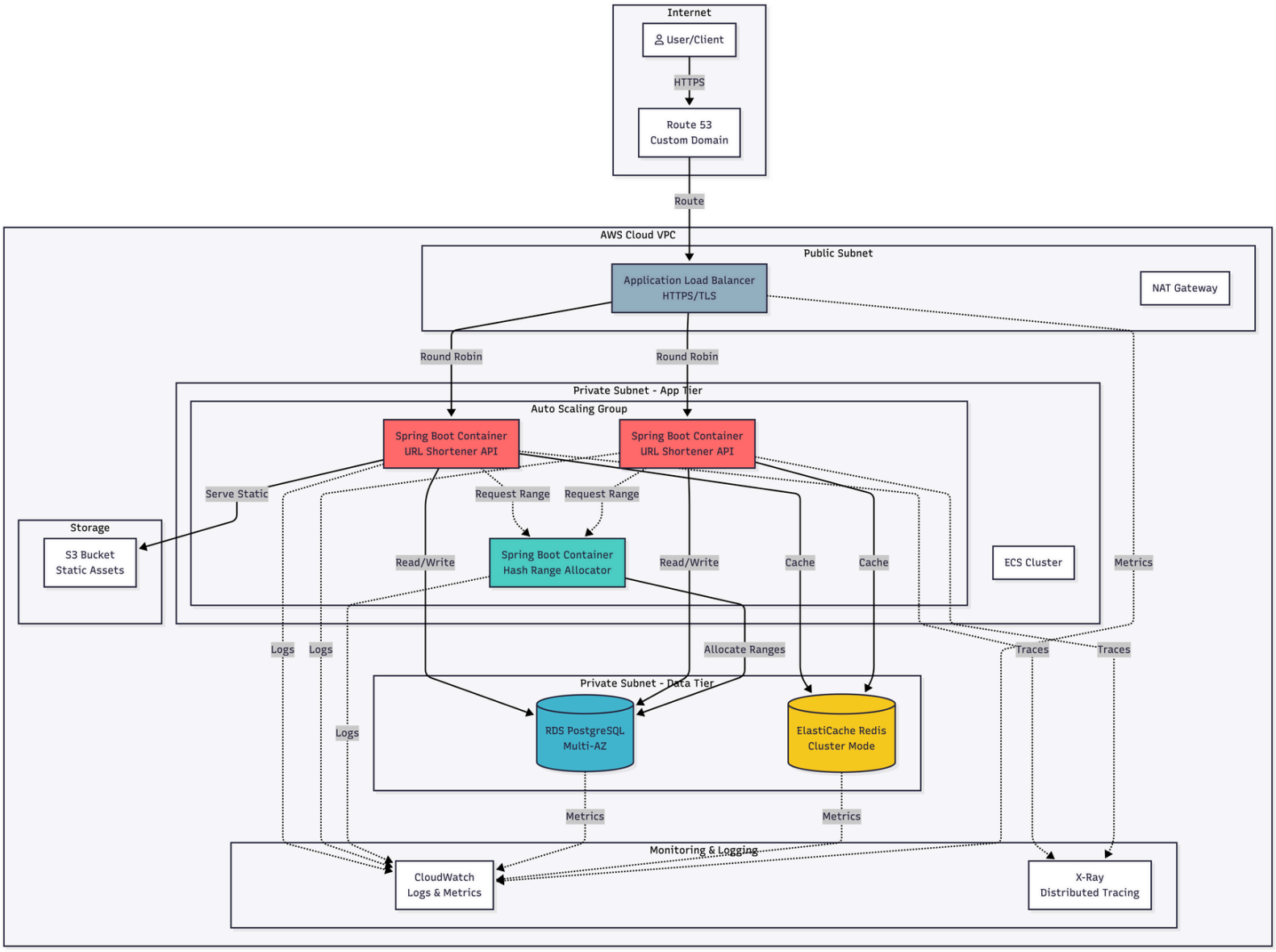
SEQUENCE DIAGRAM – URL REDIRECT



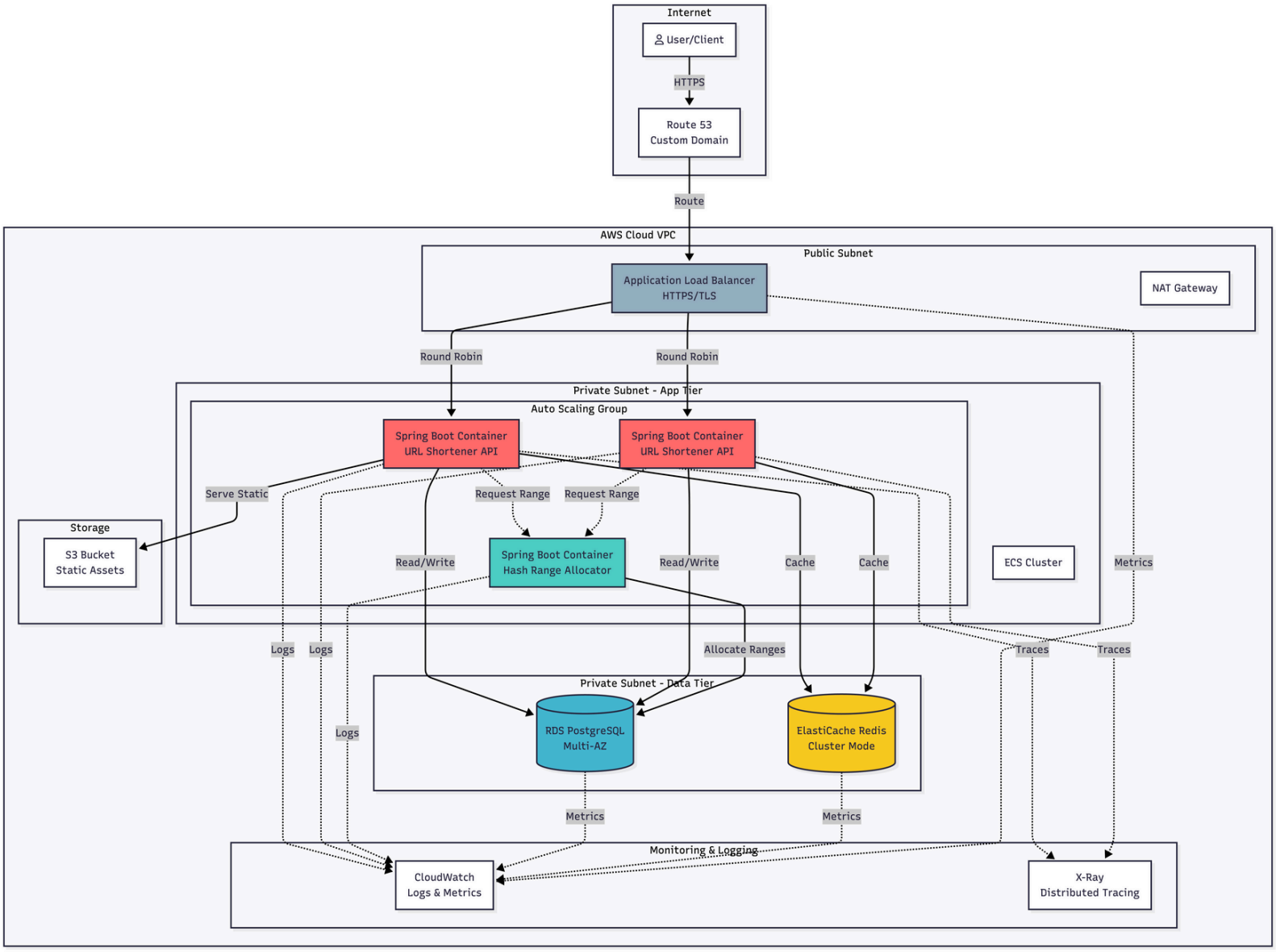
SEQUENCE DIAGRAM – USER MANAGES URLS



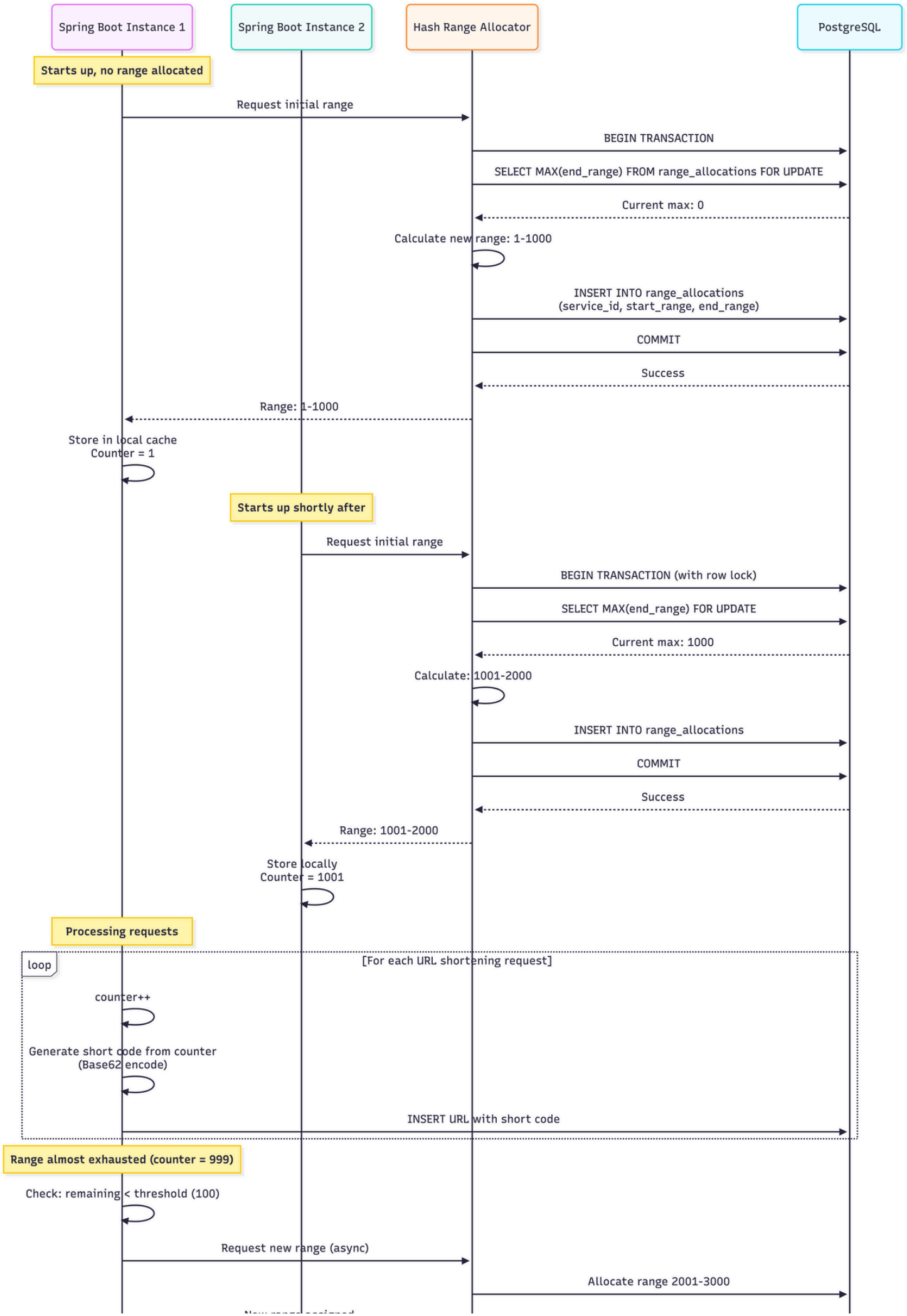
AWS ARCHITECTURE DIAGRAM

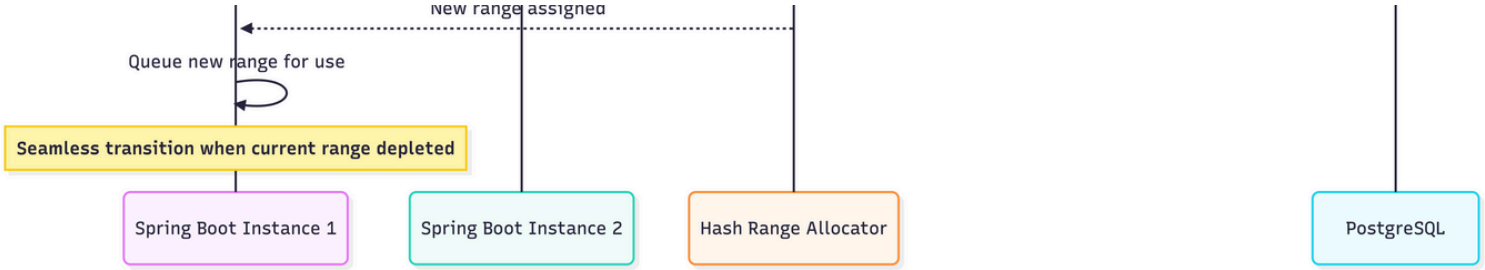


HASH RANGE ALLOCATOR – DETAILED DESIGN

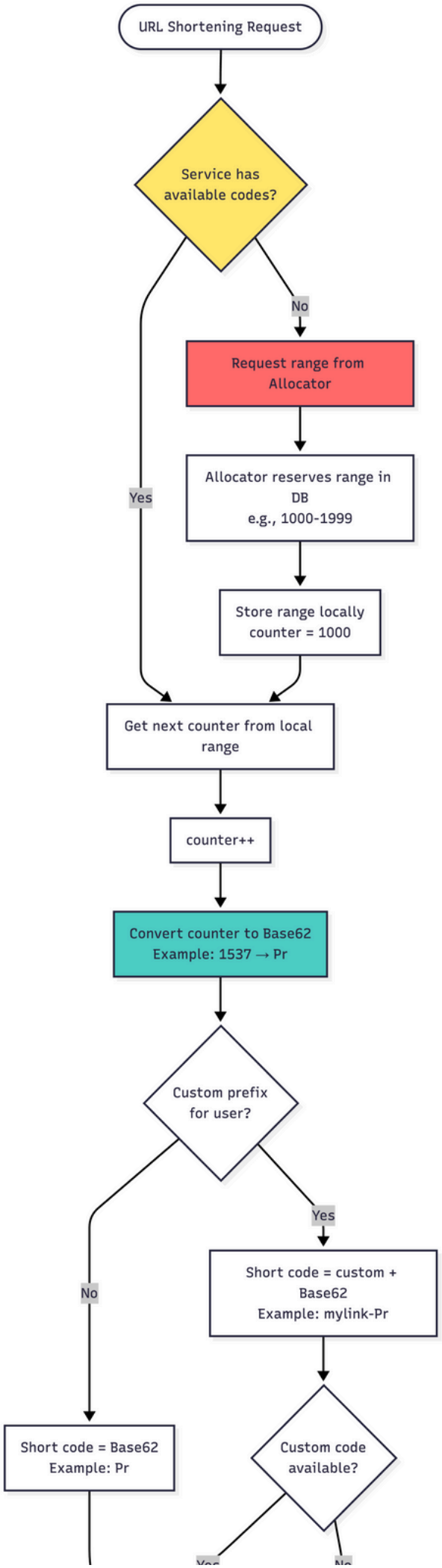


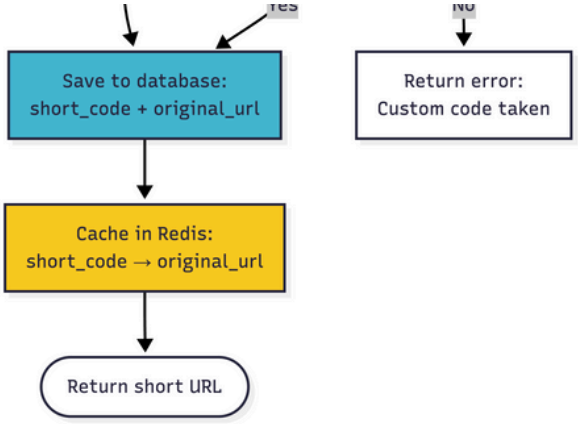
HASH RANGE ALLOCATION SEQUENCE


Spring Boot Instance 1
Spring Boot Instance 2
Hash Range Allocator
PostgreSQL
Starts up, no range allocated
Request initial range
BEGIN TRANSACTION
SELECT MAX(end_range) FROM range_allocations FOR UPDATE
Current max: 0
Calculate new range: 1-1000
INSERT INTO range_allocations
(service_id, start_range, end_range)
COMMIT
Success
Range: 1-1000
Store in local cache
Counter = 1
Starts up shortly after
Request initial range
BEGIN TRANSACTION (with row lock)
SELECT MAX(end_range) FOR UPDATE
Current max: 1000
Calculate: 1001-2000
INSERT INTO range_allocations
COMMIT
Success
Range: 1001-2000
Store locally
Counter = 1001
Processing requests
loop
[For each URL shortening request]
counter++
Generate short code from counter
(Base62 encode)
INSERT URL with short code
Range almost exhausted (counter = 999)
Check: remaining < threshold (100)
Request new range (async)
Allocate range 2001-3000


New range assigned
Queue new range for use
Seamless transition when current range depleted
Spring Boot Instance 1
Spring Boot Instance 2
Hash Range Allocator
PostgreSQL

SHORT CODE GENERATION ALGORITHM


URL Shortening Request
Service has available codes?
No
Request range from Allocator
Allocator reserves range in DB
e.g., 1000-1999
Store range locally
counter = 1000
Yes
Get next counter from local range
counter++
Convert counter to Base62
Example: 1537 → Pr
Custom prefix for user?
Yes
Short code = custom + Base62
Example: mylink-Pr
No
Short code = Base62
Example: Pr
Custom code available?


Yes
No
Save to database:
short_code + original_url
Return error:
Custom code taken
Cache in Redis:
short_code → original_url
Return short URL

DATABASE DESIGN – RANGE ALLOCATIONS

RANGE_ALLOCATIONS			
bigserial	id	PK	
varchar	service_instance_id	UK	Container/Instance ID
bigint	start_range		Start of allocated range
bigint	end_range		End of allocated range (inclusive)
timestamp	allocated_at		DEFAULT NOW()
timestamp	expires_at		Optional: for cleanup
varchar	status		ACTIVE, EXHAUSTED, EXPIRED

GLOBAL_COUNTER			
int	id	PK	Always 1 (single row table)
bigint	current_counter		Atomic counter for next range
bigint	range_size		Default: 1000
timestamp	last_updated		