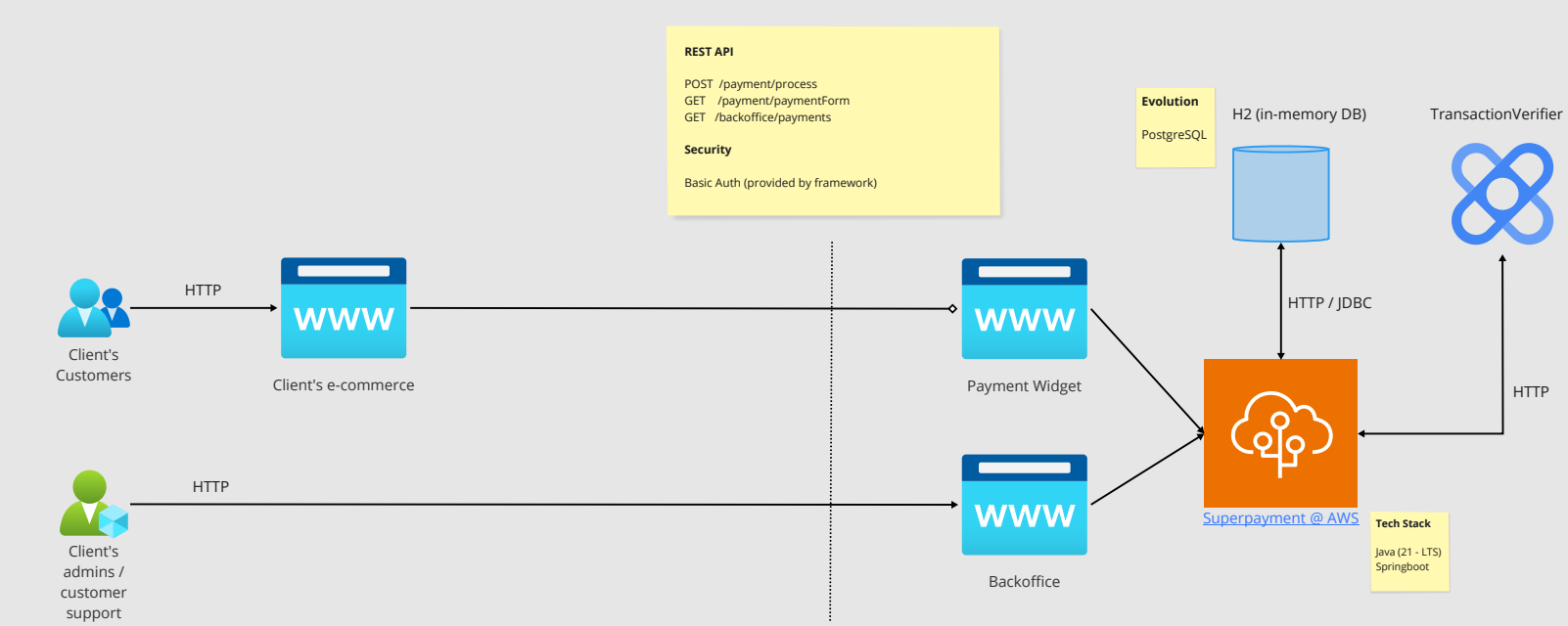
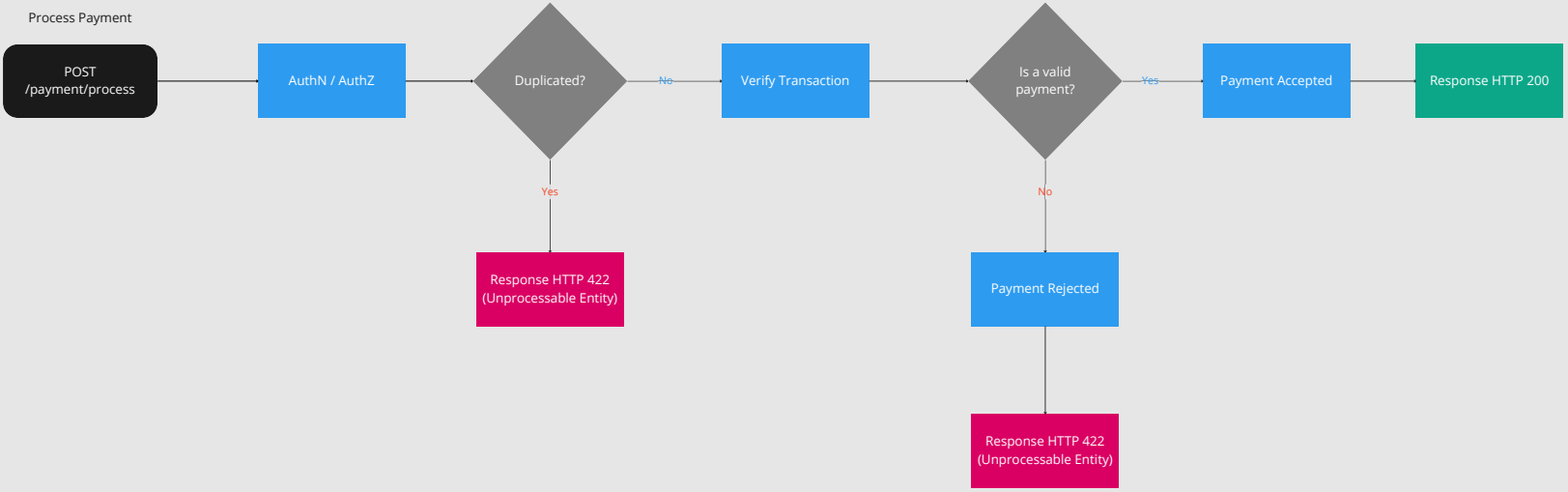
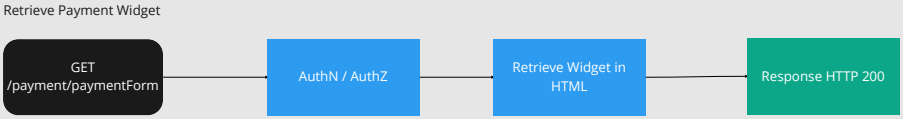
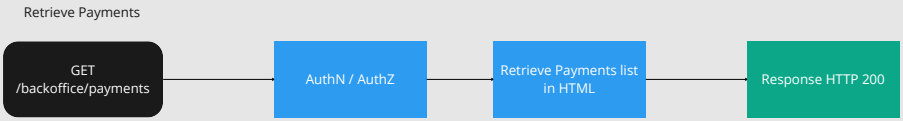

REST API
POST /payment/process
GET /payment/paymentForm
GET /backoffice/payments
Security
Basic Auth (provided by framework)
Evolution
PostgreSQL
H2 (in-memory DB)
TransactionVerifier
HTTP
Client's Customers
Client's e-commerce
Payment Widget
HTTP / JDBC
HTTP
HTTP
Client's admins / customer support
Backoffice
Superpayment @ AWS
Tech Stack
Java (21 - LTS)
Springboot

Assuming AuthN / AuthZ went OK

Security
Basic Auth (provided by framework)








Product
Logic for managing the company / customer / tenant
Get Payments in the backoffice should filter / retrieved by company ID
In a multi-tenancy env, this should be enforced internally by the Authorization capabilities
A Voucher could have max & min amount & expiration date
Voucher code and ID in prod should be two (2) different things for security, personalization and cognitive load reason around the UUID format
Should code + voucher + company be unique? This allows the voucher to be applied multiple times for the same company if a new code is generated
Plan B -> Voucher + Company
Code Refactor
Enhance security with:
- HTTPS / SSL Certificate
- CSRF
- OAuth 2.0 / JWT vs. API keys
- IndexedDB
In Payment service, limiting the max page size
The UX sucks, redo it.
In payment.html, extract the CSS and in genaral creating an architecture for styles and scripts
Extract the timezone from the payment http request for localisation and auditing purposes
Implementing flyway, for DB schema migration / evolution, before Posgres
Refactor PaymentService tests, improve interaction with the Repository.
Double check Services' validations
No Updates at DB level, just inserts and deletes.
Soft deletion, with a maintenance job
Backoffice:
- Allowing customers to set the page size
- Table's fixed size
- Timezone display in the cloud server
Include the company_id in every entity (denormalization), for multi-tenancy evolution
Infra
Automate AWS / cloud deployment
Observability (metrics, logs, tracing)
Set-up Postgres, after Flyway implementation
Docker image to ECR, generated by GH actions
Code Explore
Spock for testing
Use a lightweight JDBC lib, like jdbi instead of JPA
Rate liming for SpringBoot
Implementing an in-house pagination model
Auth0
Doc
Instructions for local deployment in Readme
CI / CD diagram here
Export postman collection and version control it
Swagger for documenting the API.
Notes

Company

- Primary Key: ID
- Soft-deletion ready. Compaction job required.

ID	NAME	DELETED
VARCHAR(255) - UUID	VARCHAR(255)	Boolean

Payment

- Primary Key: ID
- Indexes:
* VOUCHER (non unique)
* VOUCHER, CODE, COMPANY_ID (unique)
- Optimistic locking (VERSION)
- Soft-deletion ready. Compaction job required.

ID	COMPANY_ID	CODE	ORDER_AMOUNT	STATUS	VOUCHER	VERSION	DELETED	INSERT_TIMESTAMP	LAST_UPDATE_TIMESTAMP
VARCHAR(255) - UUID	VARCHAR(255) - UUID	VARCHAR(255)	NUMERIC(38,2)	TINYINT (In app, PENDING, REJECTED, ACCEPTED)	VARCHAR(255)	INTEGER	Boolean	TIMESTAMP(6) - UTC	TIMESTAMP(6) - UTC

Version in DB, for concurrency -> For transaction deduplication, optimistic locking is in place.