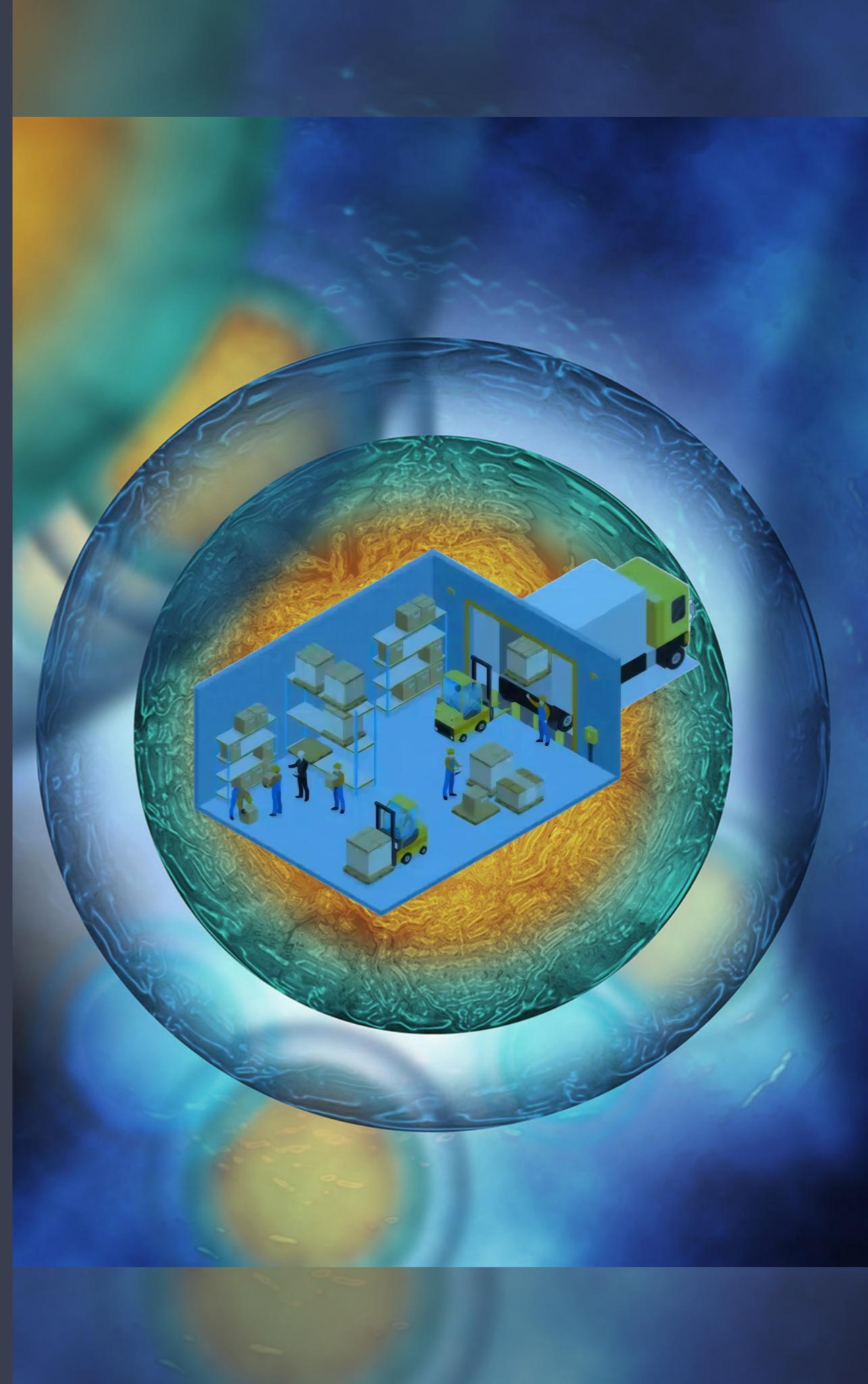


SMART INVENTORY SYSTEM

WEEK 1

Serve static data

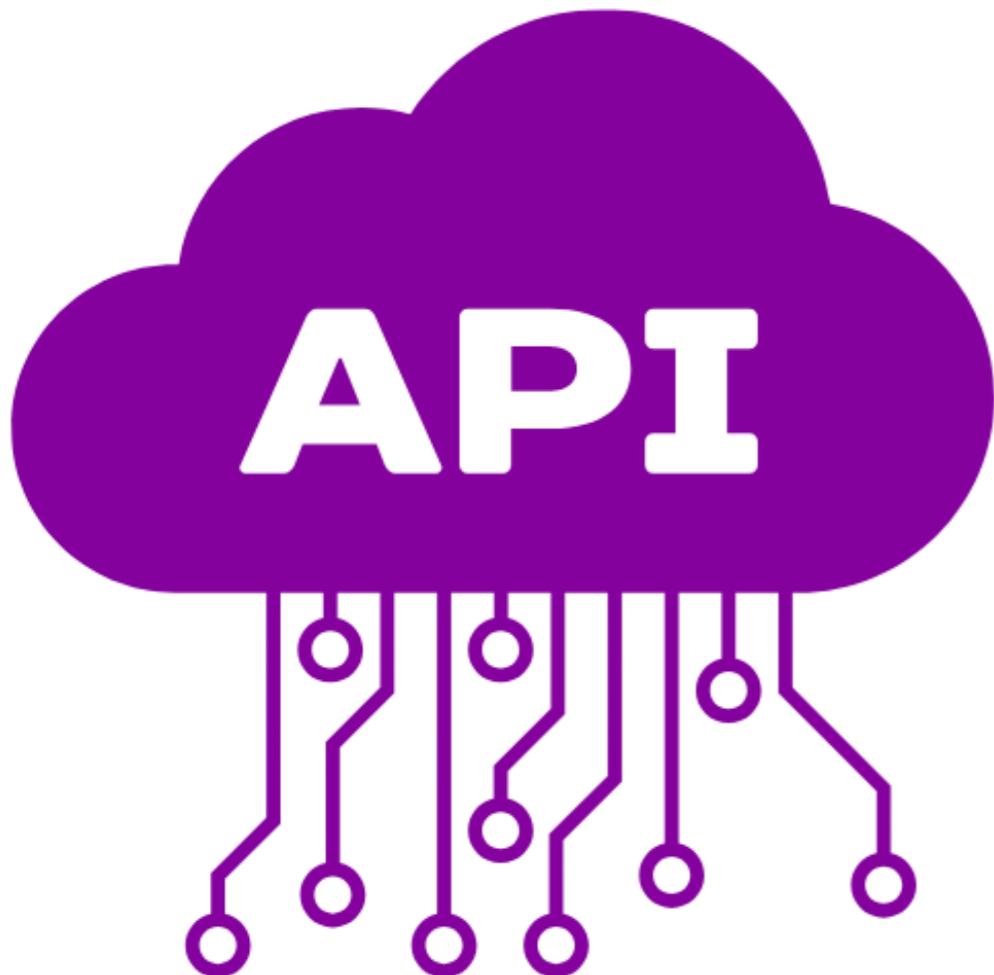


THÈME DU PROJET: SERVIR DES DONNÉES STATIQUES (PRODUCTS, ORDERS)

Livrables :

- Serveur HTTP Node sans framework, port **configurable**.
- Endpoints lecture seule sur fichiers JSON (products.json, orders.json).
- Routage propre, erreurs (200/201/400/404/500), réponses JSON.
- Logging minimal via EventEmitter.
- Fichier README (Markdown) + scripts npm + exemples curl.
- Arborescence claire (server / router / controllers / services / utils).

```
smartinventory/
├── data/
│   ├── products.json
│   └── orders.json
└── src/
    ├── server.js
    ├── router.js
    ├── controllers/
    │   ├── productsController.js
    │   └── ordersController.js
    ├── services/
    │   ├── productsService.js
    │   └── ordersService.js
    └── utils/
        ├── sendJson.js
        ├── parseQuery.js
        └── logger.js          // EventEmitter (request:received / response:sent)
├── .env.sample           // PORT=3000
├── package.json
└── README.md
```



Produits :

- GET /api/products
Query : q, category, minPrice, maxPrice, inStock, page, limit
- GET /api/products/:id
- GET /api/products/sku/:sku

Commandes :

- GET /api/orders
Query : status, from, to, page, limit
- GET /api/orders/:id
- GET /api/orders/number/:orderNumber

Santé :

- GET /health → { status:"ok", uptime, timestamp }
- **Sinon → 404** { error:"Not Found" }

Pagination :

total, page, pages ;
400 si query invalide ;
500 si lecture JSON échoue.

Gestion du cycle de vie & scripts npm (sans nodemon)

```
{  
  "name": "smartinventory",  
  "version": "1.0.0",  
  "main": "src/server.js",  
  "scripts": {  
    "start": "node src/server.js",  
    "dev": "node src/server.js"  
  },  
  "engines": { "node": ">=18 <23" }  
}
```

package.json (exemple minimal)

Utilisez *npm run dev* (ou *npm start*)

Interdits : Express et lib HTTP tierces ; Pas d'écriture disque (lecture seule).

Réponses : Content-Type: application/json; charset=utf-8

Parsing URL : url.parse(req.url, true) → pathname, query

Routage : dans router.js (éviter les énormes if/else non lisibles)

Services : lecture data/*.json (cache mémoire acceptable)

Logs : logger.js (extends EventEmitter)

- Émettre `request:received { method, url }`
- Émettre `response:sent { statusCode, route }`

Erreurs :

- 400 : query invalide (minPrice>maxPrice, dates invalides...)
- 404 : ressource absente
- 500 : problème lecture/parse JSON

Config : .env.sample avec PORT=3000 (ou défaut 3000)

SCÉNARIOS DE TEST

-> Produits

```
curl "http://localhost:3000/api/products?category=tools&minPrice=10&maxPrice=100&page=2&limit=5"  
curl "http://localhost:3000/api/products/sku/SKU-001"  
curl "http://localhost:3000/api/products/123"
```

-> Commandes

```
curl "http://localhost:3000/api/orders?status=paid&from=2024-01-01&to=2024-12-31&limit=20"  
curl "http://localhost:3000/api/orders/number/ORD-2025-0007"
```

-> Santé

```
curl "http://localhost:3000/health"
```