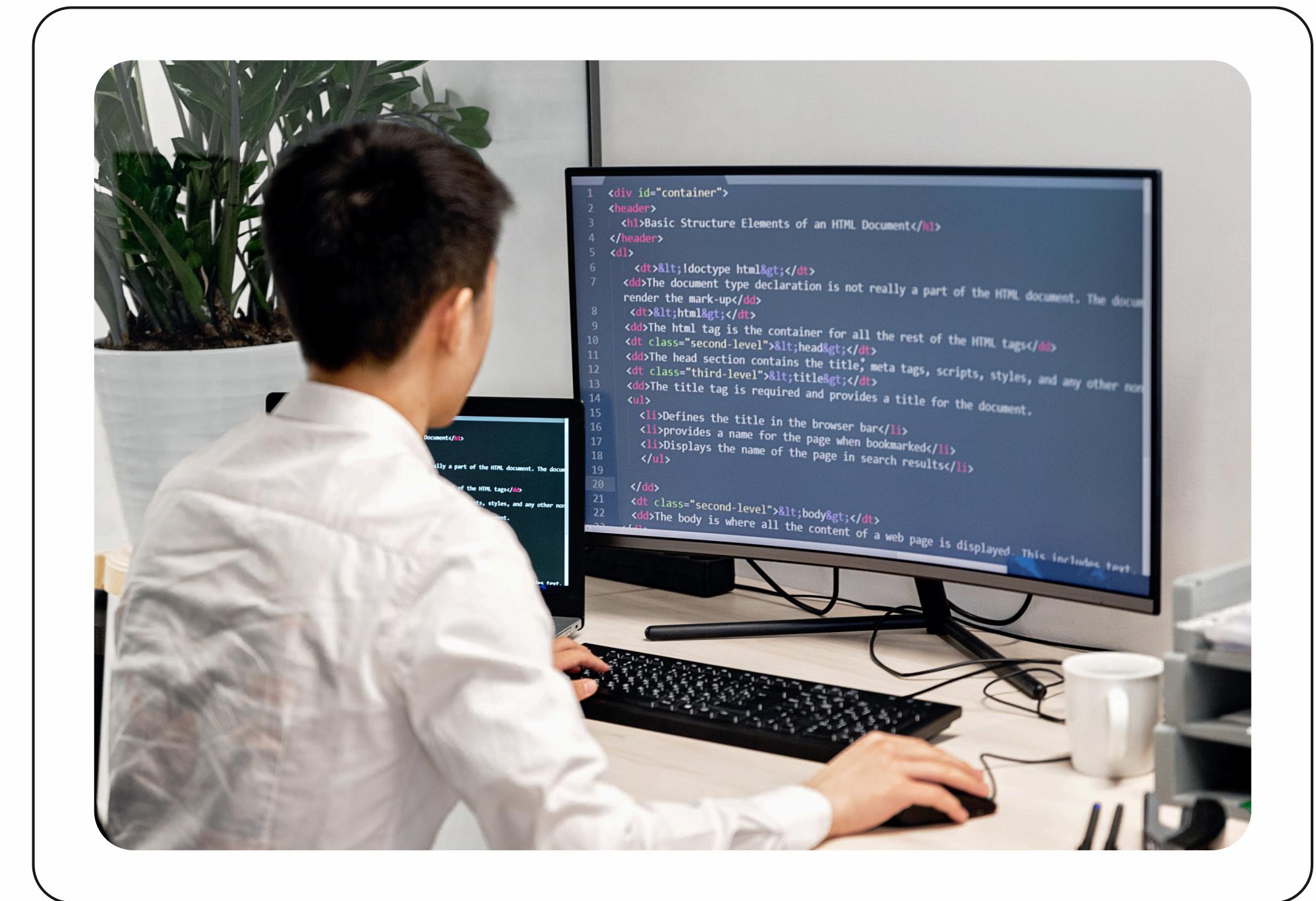


SHOPPING CART



Fanelesibonge Mbuyazi



OVERVIEW



```
AreaSeries, Trendlines }  
import { EmaIndicator, RsiIndicator }
```

Task

Improve a basic shopping cart implementation

Focus

Readability, maintainability, correctness

Out of scope

Database, testing, security

Approach

Refactor into clean layered architecture

OVERVIEW

The original code functionality:

- Add items (name, price, quantity) to a cart
- Track multiple carts using unique cart IDs
- Calculate the total cost of all items



FOCUS: Readability, maintainability, correctness

1. Separated code into Controller, Service, Model, DTO layers

2. Introduced strong typing with Cart & CartItem classes

4. Added validation and structured error handling

3. Replaced doubles with BigDecimal for monetary accuracy

5. Designed RESTful APIs with proper HTTP responses

APPROACH

```
    --> = Status(
    Status_id=data.id, name=data.name, desc=
)
statuses[status.name] = status
```

Java spring boot architecture

```
com.example.projectname
  └ controller // API endpoints (REST)
    └ service // Business logic
    └ repository // Database layer (Spring Data JPA)
    └ model // Entity/DTO classes
    └ config // Custom configurations (e.g. security, CORS)
    └ exception // Custom exceptions & handlers
    └ security // JWT/OAuth2 setup
    └ utils // Helper classes (converters, validators)
```

ARCHITECTURE

```
status = Status(
    status_id=data.id, name=data.name, ...
)
statuses[status.name] = status
```

- Benefits
- Clear separation of concerns (**maintainability & correctness**)
- Easy onboarding for new developers (**readability**)
- Scalable for microservices or monoliths
- Cleaner dependency injection and testing

Refactored the code into a layered architecture
(Controller, Service, Model, DTO)

```
shopping-cart/
├── controller/          # REST controllers
├── service/             # Business logic
├── model/               # Domain models (Cart, CartItem)
└── exception/           # Custom exceptions
├── dto/                 # Response DTOs
├── resources/            # Application resources
└── ShopApplication.java # Spring Boot entry point
```

ARCHITECTURE

```
status = Status(
    status_id=data.id, name=data.name, desc=data.desc)
statuses[status.name] = status
```

- **Controller Layer** – Handles HTTP requests (**ShoppingCartController**)
- **Service Layer** – Core business logic (**ShoppingCartService**)
- **Model Layer** – Domain objects (**Cart, CartItem**)
- **DTO Layer** – API responses (**CartResponse**)
- **Exception Layer** – Centralized error handling (**CartNotFoundException**)

KEY IMPROVEMENTS

Improvements

- Clean separation of concerns
- Immutability with Java records
- RESTful API design with proper HTTP codes
- Centralized exception handling
- Extensible for future features (e.g., remove items, discounts)



```
import { StockChartComponent, StockChart } from '@sanlam/stockchart';
import { MultiTip, RangeTooltip, Crosshair, AreaSeries, Trendlines } from '@sanlam/stockchart';
import { EmaIndicator, BollingerBands, AccumulationDistribution, Candles, MovingAverages } from '@sanlam/stockchart';
import { SampleBase } from '@sanlam/stockchart';
export class Default extends Component {
  constructor() {
    super();
    this.state = {
      chart: null,
      data: [
        {
          date: '2023-01-01',
          open: 100,
          close: 105,
          low: 98,
          high: 108,
          volume: 1000000
        },
        {
          date: '2023-01-02',
          open: 105,
          close: 110,
          low: 102,
          high: 112,
          volume: 1200000
        },
        {
          date: '2023-01-03',
          open: 110,
          close: 115,
          low: 108,
          high: 118,
          volume: 1100000
        },
        {
          date: '2023-01-04',
          open: 115,
          close: 120,
          low: 112,
          high: 122,
          volume: 1300000
        },
        {
          date: '2023-01-05',
          open: 120,
          close: 125,
          low: 118,
          high: 128,
          volume: 1400000
        }
      ]
    };
  }

  render() {
    return (
      <div>
        <StockChart
          data={this.state.data}
          chartType="Candlestick"
          crosshair="true"
          tooltip="true"
          trendlines="true"
          area="true"
          ema="true"
          bollinger="true"
          accumulation="true"
          candles="true"
          movingAverages="true"
        />
      </div>
    );
  }
}
```

Key Changes

- **Endpoints renamed:**
 - From addItem → /shop/{cartId}/addItems
 - From getTotal → /shop/{cartId}/total
- Path Variables introduced: cartId is now part of the URL path (RESTful style).
- **Responses changed:**
 - Originally: plain strings ("Item added. Total: X")
 - Now: structured JSON DTOs (CartResponse).
- **HTTP status codes:**
 - 201 Created for adding items
 - 200 OK for retrieving totals
 - 400 Bad Request for invalid inputs
 - 404 Not Found if the cart doesn't exist

FUTURE IMPROVEMENTS

Improvements

- Add persistence (database integration)
- Introduce unit and integration testing
- Add Swagger/OpenAPI documentation
- Implement discount/promotion engine
- Improve concurrency handling further





THANK YOU

Get In Touch



+27 67 229 8553



fanelesibongembuyazi0@gmail.com



<https://www.linkedin.com/in/fanelesi-bonge-mbuyazi-036934193/>