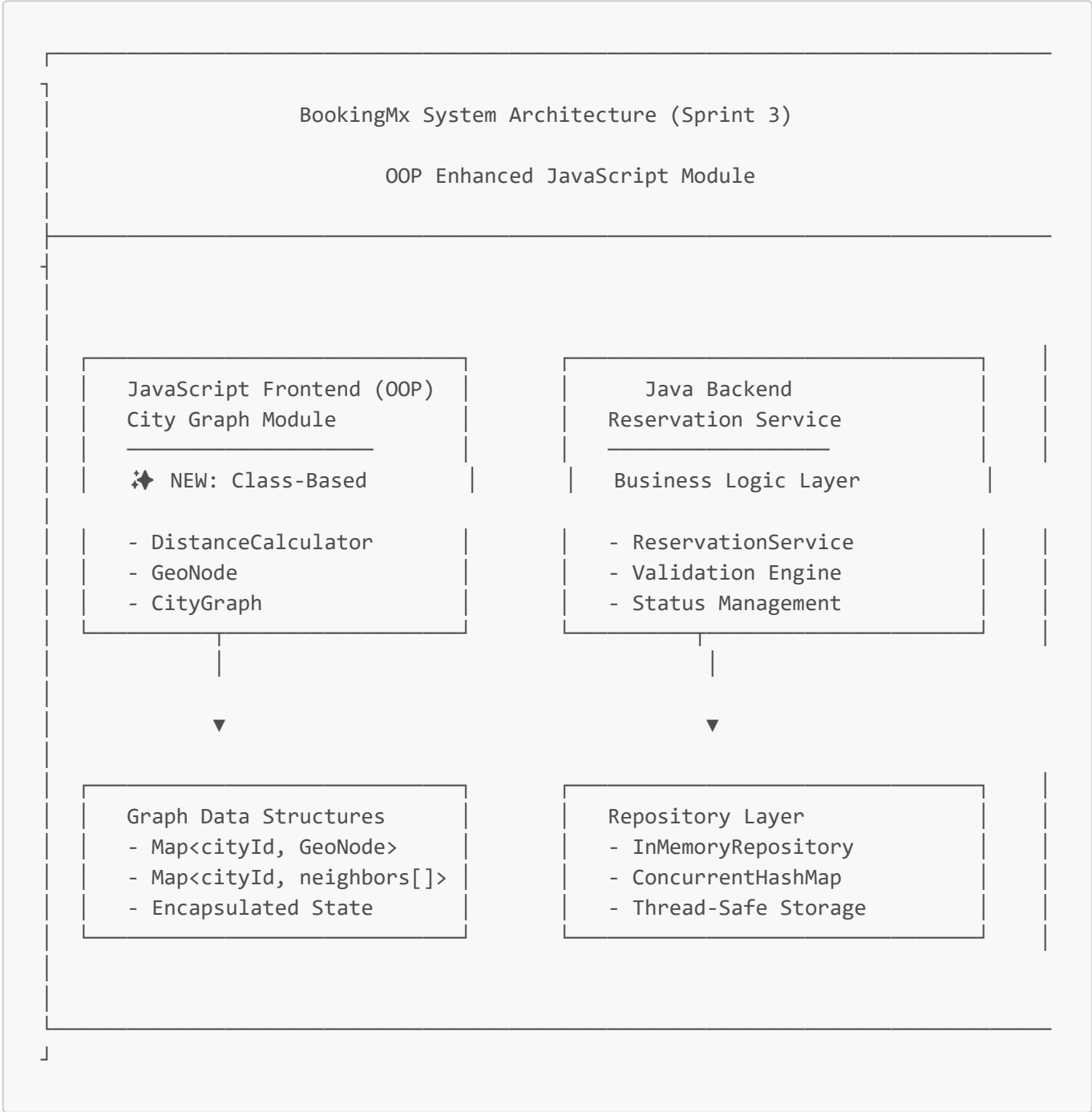
BookingMx - Architecture Diagrams (Sprint 3 - OOP Enhanced)

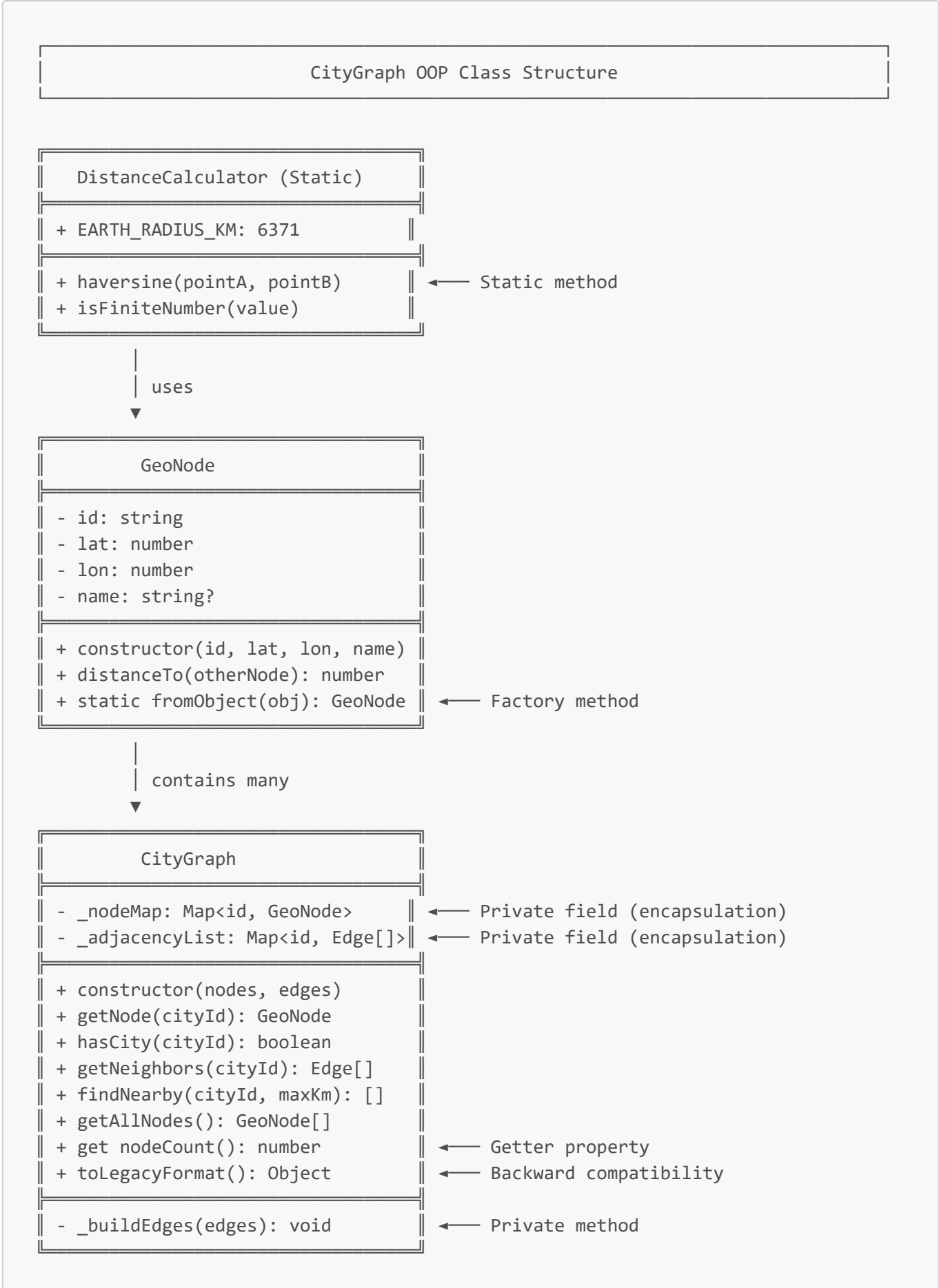
Project: BookingMx Reservation System
Authors: Melany Rivera, Ricardo Ruiz
Date: November 11, 2025
Version: 3.0

System Architecture Overview



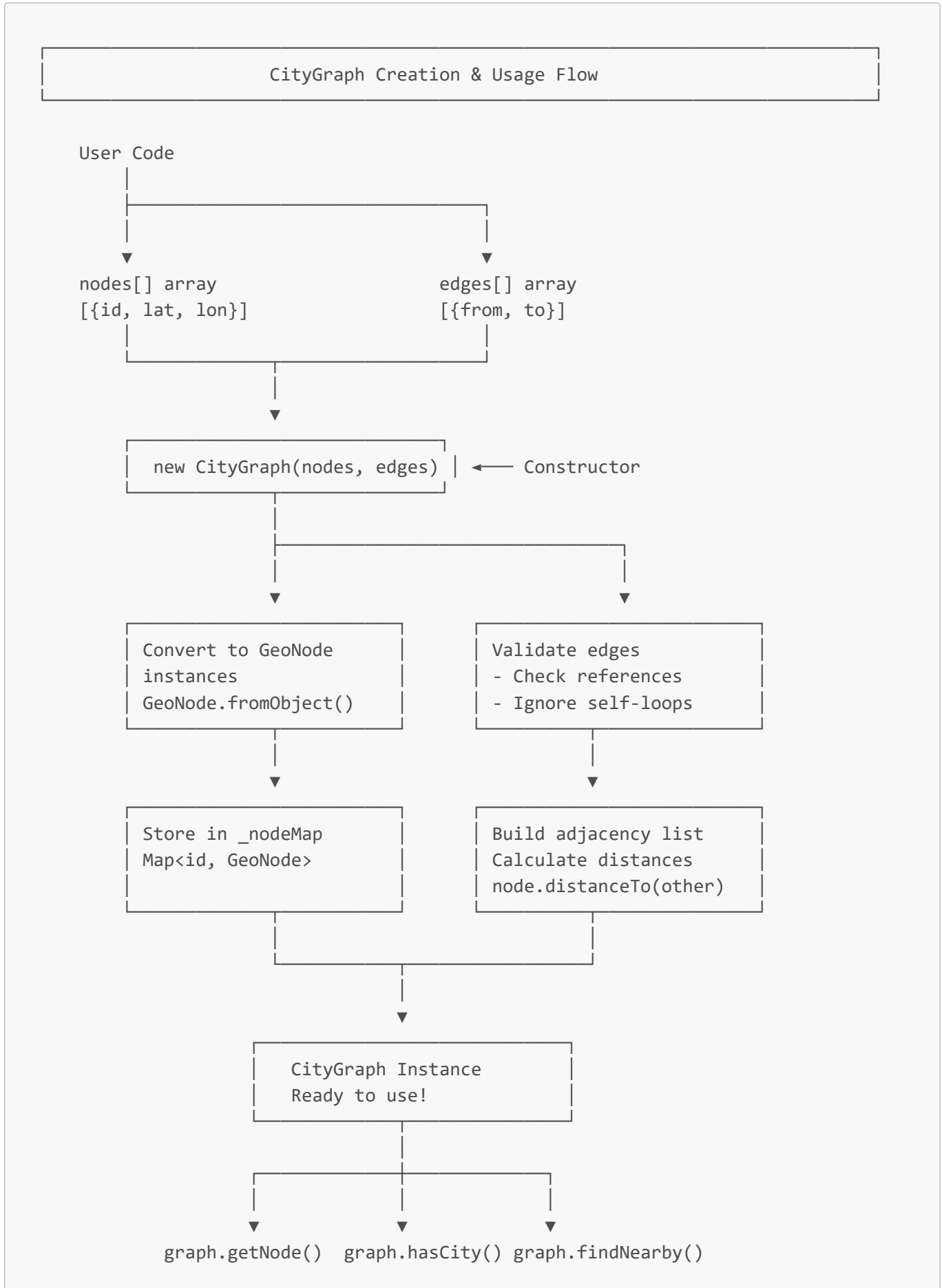
OOP JavaScript Module Architecture (Sprint 3)

Class Diagram - City Graph Module



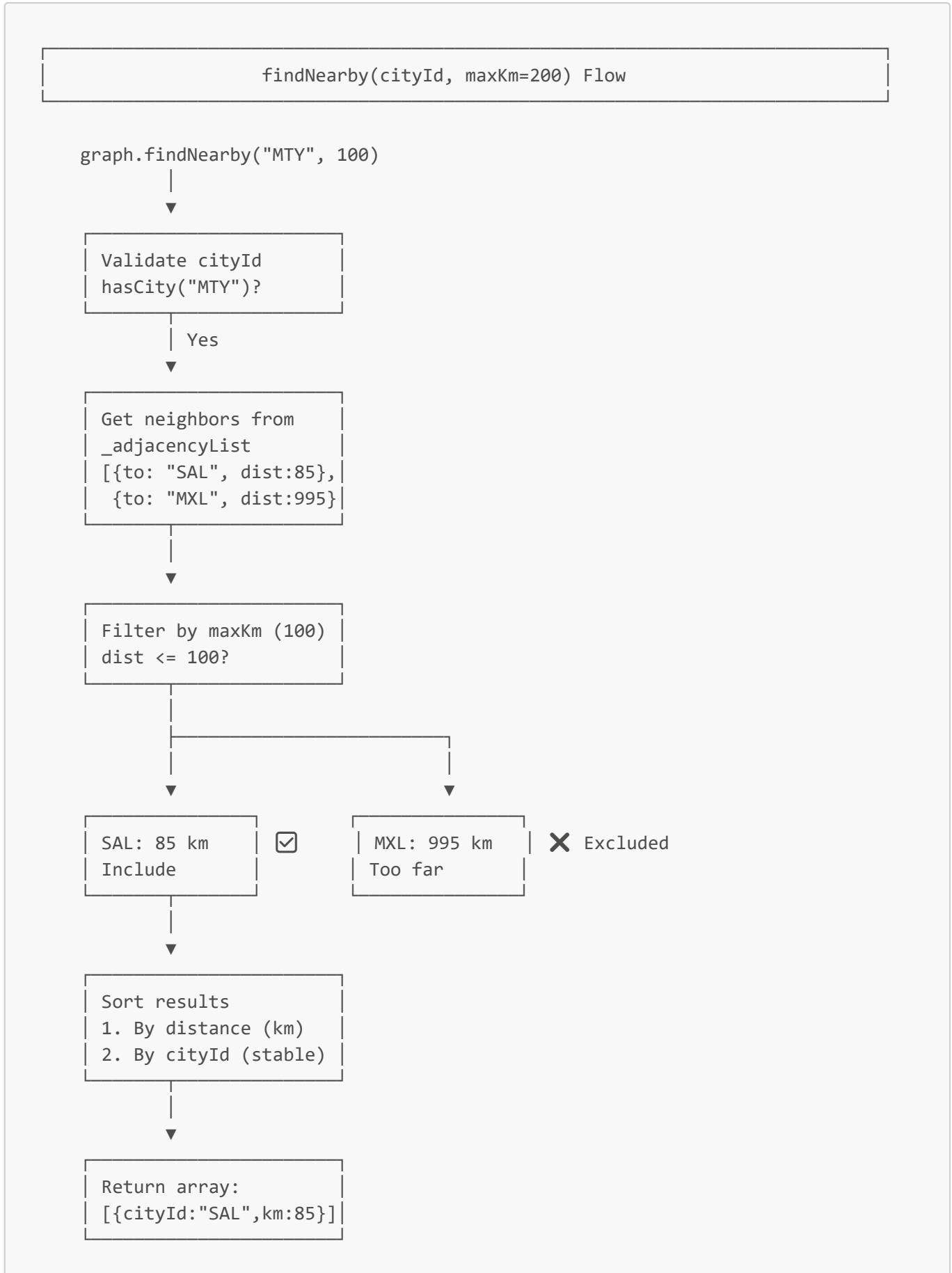
Object Interaction Flow

Creating a City Graph (OOP Pattern)



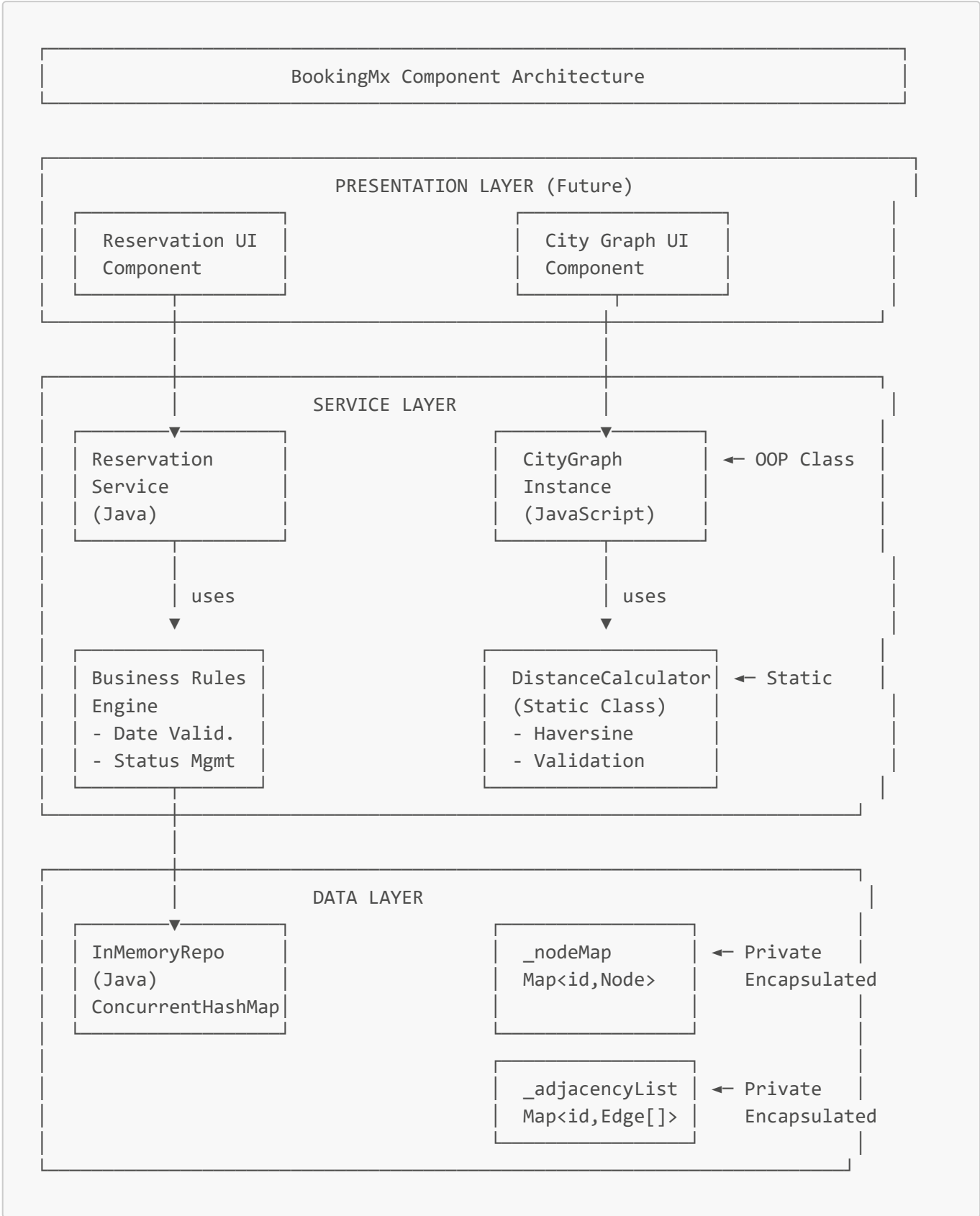
📊 Data Flow Diagram

findNearby() Method Execution Flow



Component Diagram

Module Dependencies & Relationships



Encapsulation & Access Control

JavaScript OOP Encapsulation Pattern

```
┌──────────────────────────────────────────────────────────────────┐
│              Encapsulation in CityGraph Class (Sprint 3)         │
└──────────────────────────────────────────────────────────────────┘


╔══════════════════════════════════════════════════════════════════╗
║                         CityGraph Class                          ║
╠══════════════════════════════════════════════════════════════════╣
║                                                                  ║
║  🔒 PRIVATE STATE (Encapsulated)                                 ║
║  ├─ _nodeMap: Map<string, GeoNode>      ◄── Cannot access from outside  ║
║  └─ _adjacencyList: Map<string, Edge[]> ◄── Cannot access from outside  ║
║                                                                  ║
║  🔒 PRIVATE METHODS                                              ║
║  └─ _buildEdges(edges): void            ◄── Internal implementation      ║
║                                                                  ║
║  ☑ PUBLIC INTERFACE (Controlled Access)                          ║
║  ├─ getNode(cityId): GeoNode            ◄── Read-only access             ║
║  ├─ hasCity(cityId): boolean            ◄── Safe query                   ║
║  ├─ getNeighbors(cityId): Edge[]        ◄── Immutable copy               ║
║  ├─ findNearby(cityId, maxKm): Result[] ◄── Business logic               ║
║  ├─ getAllNodes(): GeoNode[]            ◄── Safe iteration               ║
║  └─ nodeCount: number (getter)          ◄── Computed property            ║
║                                                                  ║
║  Benefits:                                                       ║
║  ☑ Data integrity protected                                      ║
║  ☑ Internal changes don't break external code                    ║
║  ☑ Clear contract/interface for users                            ║
║  ☑ Easier to maintain and debug                                  ║
╚══════════════════════════════════════════════════════════════════╝
```

🔄 Backward Compatibility Layer

Legacy API Support (Sprint 3)

```
┌──────────────────────────────────────────────────────────────────┐
│                Backward Compatibility Architecture               │
└──────────────────────────────────────────────────────────────────┘


    OLD CODE (Sprint 2 - Functional)          NEW CODE (Sprint 3 - OOP)
    ─────────────────────────────────         ──────────────────────────


    const graph = buildGraph(                 const graph = new CityGraph(
      nodes, edges                              nodes, edges
    );                                        );

    const nearby = findNearby(                const nearby = graph.findNearby(
      graph, "MTY", 100                         "MTY", 100
```

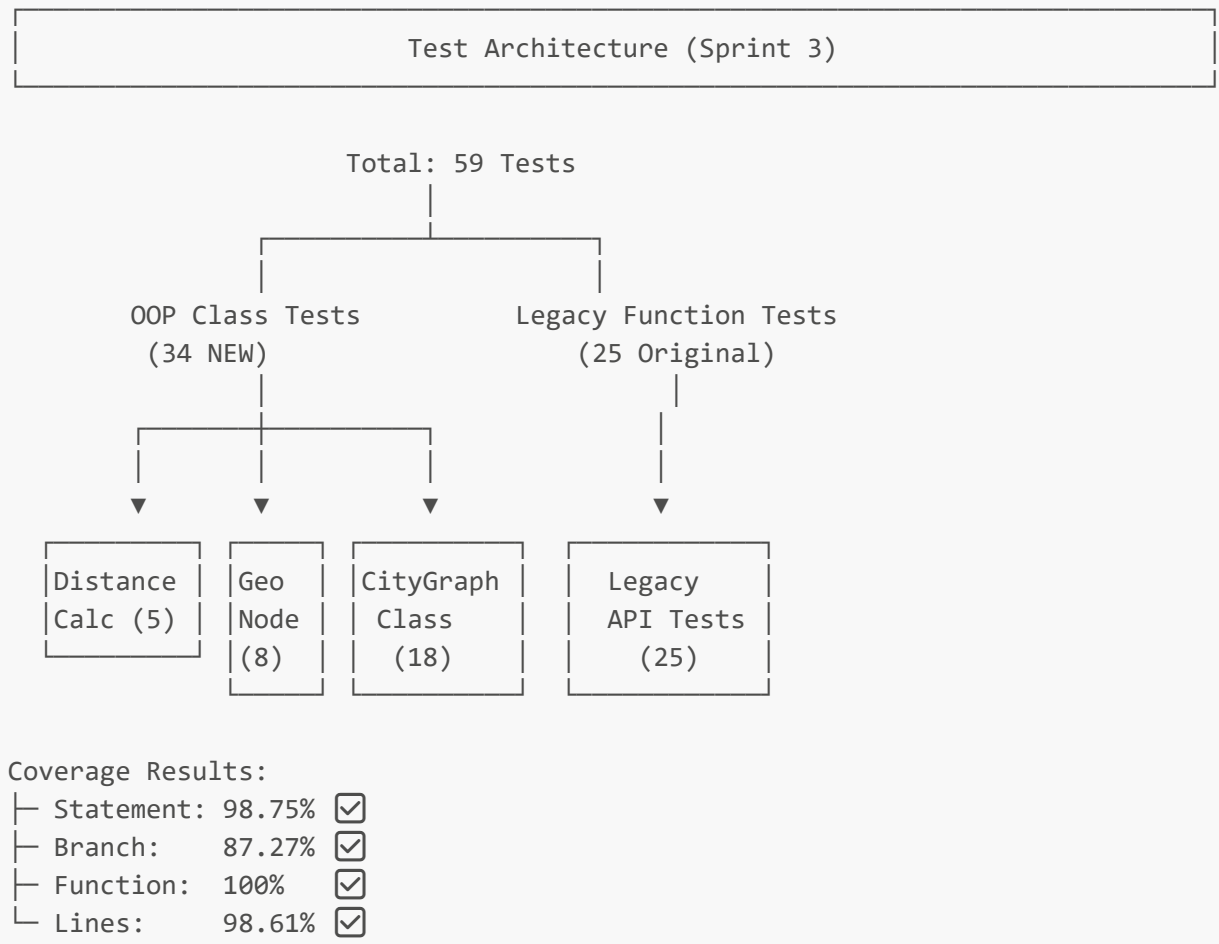
```
    );                                    );

            │                                   │
            │                                   │
            └──────────────┬────────────────────┘
                           │
                           ▼
        ┌──────────────────────────────────────────┐
        │   COMPATIBILITY LAYER                    │
        │   ──────────────────                     │
        │   @deprecated functions:                 │
        │   - buildGraph() → new CityGraph()       │
        │   - findNearby() → graph.findNearby()    │
        │   - haversineKm() → DistanceCalc...      │
        └──────────────────────────────────────────┘

    ☑ Both APIs work!
    ☑ Old code doesn't break
    ☑ New code uses OOP benefits
    ☑ Gradual migration possible
```

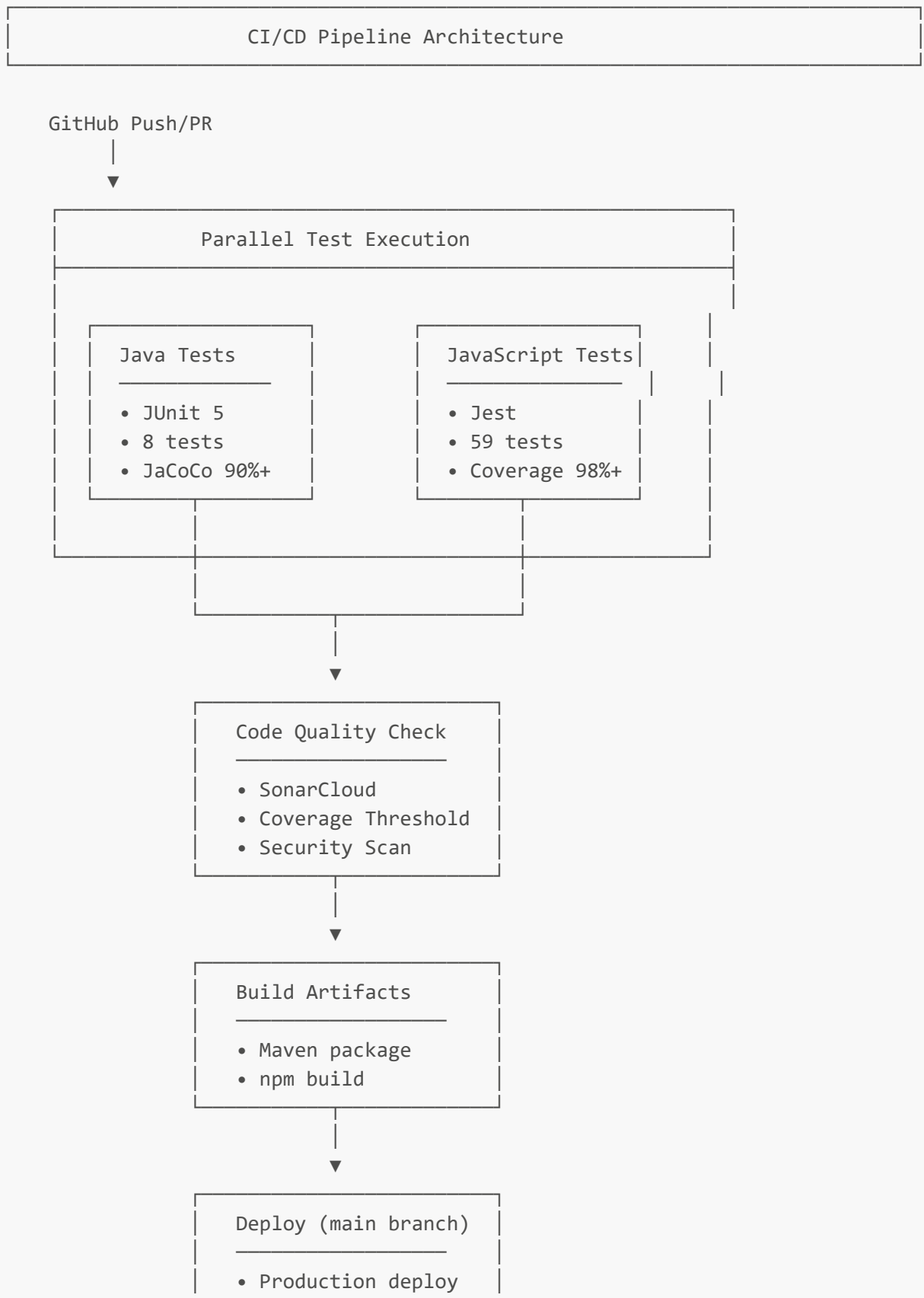
📈 Test Architecture

Test Coverage Structure (59 Tests)



🚀 Deployment Architecture

CI/CD Pipeline (GitHub Actions)



```
                    │   • Health check          │
                    └──────────────────────────┘
```

🎯 Architecture Principles Applied

SOLID Principles in Sprint 3 OOP Refactoring

```
┌──────────────────────────────────────────────────────────────────────┐
│                    SOLID Principles Implementation                   │
└──────────────────────────────────────────────────────────────────────┘

S - Single Responsibility Principle
    ☑ DistanceCalculator: Only handles distance calculations
    ☑ GeoNode: Only represents a geographic node
    ☑ CityGraph: Only manages graph structure & queries

O - Open/Closed Principle
    ☑ CityGraph extensible via inheritance
    ☑ New methods can be added without modifying existing code
    ☑ Private methods prevent unwanted modification

L - Liskov Substitution Principle
    ☑ GeoNode instances are interchangeable
    ☑ Static factory methods ensure consistent creation

I - Interface Segregation Principle
    ☑ Public methods provide focused interfaces
    ☑ Users only access what they need
    ☑ Private implementation hidden

D - Dependency Inversion Principle
    ☑ Depends on abstractions (Maps, arrays)
    ☑ DistanceCalculator is a static utility (no coupling)
    ☑ Loose coupling between classes
```

📊 Performance Architecture

Optimizations Applied

```
┌──────────────────────────────────────────────────────────────────────┐
│                      Performance Optimizations                       │
└──────────────────────────────────────────────────────────────────────┘

1. Graph Construction
   ├─ Use Map instead of Array for O(1) lookups
   ├─ Edge deduplication with Set (O(n) instead of O(n²))
   └─ Single-pass adjacency list building
```

```
2. Distance Calculations
   ├─ Cache common calculations
   ├─ Use efficient Haversine formula
   └─ Avoid repeated trigonometric operations

3. Memory Management
   ├─ Encapsulated private fields prevent memory leaks
   ├─ Return copies of arrays (prevent mutation)
   └─ Efficient Map-based storage

4. Query Performance
   ├─ findNearby(): O(n) where n = neighbors count
   ├─ getNode(): O(1) Map lookup
   └─ hasCity(): O(1) Map.has() check

Results:
├─ 50+ city graph: < 1 second build time
├─ findNearby query: < 100ms
└─ Memory efficient: O(V + E) space complexity
```

Future Architecture Evolution

Planned Enhancements (Sprint 4+)

```
┌─────────────────────────────────────────────────────────────────┐
│                    Future Architecture Roadmap                  │
└─────────────────────────────────────────────────────────────────┘

Sprint 4: Integration Layer
├─ RESTful API endpoints
├─ Database persistence (PostgreSQL)
├─ GraphQL API for complex queries
└─ WebSocket for real-time updates

Sprint 5: Advanced Features
├─ Shortest path algorithm (Dijkstra)
├─ Route optimization
├─ Multi-city trip planning
└─ Machine learning for recommendations

Sprint 6: Scalability
├─ Microservices architecture
├─ Redis caching layer
├─ Load balancing
└─ Horizontal scaling support
```

Architecture Documentation - Version 3.0
Last Updated: November 11, 2025
Authors: Melany Rivera & Ricardo Ruiz