Come up with a smart and scalable output schema that is future-proof. Explain why you think it is so.

1. **Data Model:**
    - Courier: Represents a courier with attributes such as ID, name, current capacity, and other relevant information.
2. **Endpoints:**

    Business endpoint that allow the backend to receive and process requests to perform crud operation on the couriers

    Kafka Topics:

    - `courier_created`: Used to publish a message when a new courier is created.
    - `courier_updated`: Used to publish a message when a courier's is updated.

    Kafka topics enable asynchronous communication and can act as event triggers for other microservices or components in the system.

3. **Message Payload:**
    - When a new courier is created, the `courier_created` message contains the courier's ID, and some extra details required from the business perspective.
    - When a courier's capacity is updated, the `courier_capacity_updated` message contains the courier's ID, the previous, new payload and the diff.

    The messages follow a structured format to ensure consistency and easy parsing.

4. **Scalability Considerations:**
    - Use a microservices architecture: Split functionality into smaller services for easier maintenance and scaling.

- Load Balancing: Implement load balancing techniques to distribute incoming requests across multiple instances of the backend.
- Database Scaling: Since we know that we need to work with transactions and avoid race conditions, we should consider using a relational database management system (RDBMS) that supports ACID (Atomicity, Consistency, Isolation, Durability) properties. ACID ensures that database transactions are processed reliably, and data integrity is maintained even in the presence of concurrent operations. Such as PostgreSQL, MySQL, among others.
- Kafka Partitions: Properly partition Kafka topics to distribute the load across multiple brokers.

5. **Error Handling and Monitoring:**
    - Implement robust error handling to handle failures gracefully and provide meaningful error messages.
    - Use logging and monitoring to track system health, performance, and potential issues.