Technical Report - **Product specification**

Interactive Earnings and Sale Software (IESS)

Course:	IES - Introdução à Engenharia de Software
Date:	Aveiro, 21th of December, 2023
Students:	107876: Gabriel Teixeira 104152: Gonçalo Abrantes 117878: Marcos Miguélez 105925: Rafael Kauati

Project abstract:	IES is an integrated system that supports the tracking and storing of various information relating to products and employees.

Table of contents:

1 Introduction

The goal of this project is to propose, conceptualize and implement a multi-layer enterprise-class application.

With this in mind, our project focuses on the monitoring of sales specifically in a point of sale setting. We aim to build an intuitive interface that would help in the tracking of data related to the Point of Sale business as well as help improve customer assistance.

2 Product concept

Vision statement

Our system aims to empower both store cashiers and managers with the tools they need to excel in their roles. The primary focus is to enhance the customer shopping experience by providing real-time product information and ensuring that customers find what they need. Additionally, the system will also use data analytics to foster innovation by delivering valuable insights for managers to make informed decisions.

Personas and Scenarios

Sarah Smith is a 40 year old diligent and experienced cashier at a busy supermarket. She is tech-savvy and has a keen eye for customer service. She takes pride in her work and enjoys assisting customers in finding the products they need. Sarah is responsible for managing customer transactions efficiently and ensuring that the checkout process is smooth and pleasant for every customer.

Goals and Needs: Sarah's primary goal is to assist customers in finding and purchasing products quickly and accurately. She needs a user-friendly system that allows her to manage the customer's cart, view product details, and complete transactions seamlessly. Sarah values a system that provides real-time information about product availability and pricing to assist customers effectively.

Alex Jones is a 54 year old experienced store manager with a strong business sense. He is responsible for overseeing the entire supermarket operation, including staff management, inventory control, and customer satisfaction. Alex is data-driven and makes strategic decisions based on sales analytics and customer behavior patterns. They aim to optimize the store's performance and enhance the overall customer experience.

Goals and Needs: Alex's primary goal is to ensure the supermarket operates efficiently, meets sales targets, and delivers exceptional customer service. He needs a system that provides detailed insights into employee performance, inventory levels, and product sales data to make informed decisions. Alex values a system that alerts him when popular products are running low in stock, enabling proactive restocking to meet customer demand.

Product requirements (User stories)

Epic 1 : Inventory Management

As Alex, I should be able to add and update products (and their information) in stock.

Epic 2 : Up to date Product Information (update pricing based on the stock market)

As Sarah, I want to view detailed product information, including current stock, pricing, and any ongoing promotions, to provide accurate information to customers and prevent selling products that are out of stock.

Note : the price of every product is periodically adjusted due the stock market variations.

Epic 3 : Streamlined Checkout Process

As Sarah, I want to be able to search for products and to add them to the customer’s cart with one click and confirm their payment efficiently, to ensure a smooth and pleasant checkout experience for every customer.

Epic 4 : Inventory supervision

As Alex,I want to receive automatic alerts when popular products reach a defined low-stock threshold, and I want to initiate restocking requests directly from the system.

Epic 5 : Sales Analytics Dashboard

As Alex, I want a dashboard displaying sales data with filters for different time periods, allowing me to view sales figures, compare performance, and identify trends.

Epic 6 : Employee Performance Tracking

As Alex, I want to access reports on cashier performance, including the number of transactions processed, transaction accuracy, and any errors or discrepancies in transactions.

3 Architecture notebook

Key requirements and constraints

<Identify issues that will drive the choices for the architecture such as: Will the system be driven by complex deployment concerns, adapting to legacy systems, or performance issues? Does it need to be robust for long-term maintenance?

1. Complex Deployment Concerns and Business Format Constraints:

- **Scalability: The architecture needs to be scalable to handle variable concurrency of data input/output, accommodating fluctuations in client frequency without compromising performance.**
- **Flexibility: The system must be adaptable to different store formats and business requirements, ensuring it can be deployed seamlessly across various store types and sizes.**
- **Interoperability: The architecture should support integration with diverse hardware components like barcode scanners and receipt printers (if available for the project) commonly found in retail environments.**

2. Performance and Data Freshness:

- **Real-Time Data Sync: Establish a robust data synchronization mechanism between the point of sale software and the company's data centers. This ensures that the system always displays updated and recent data to users, enhancing customer service and decision-making for store staff and managers.**
- **Network Optimization: Optimize network protocols and bandwidth usage to support real-time data updates without putting strain on the network infrastructure.**

3. Long-Term Maintenance and Robustness:

- **Error Handling and Logging: Implement comprehensive error handling mechanisms and detailed logging to track issues, diagnose problems, and ensure timely resolution, enhancing the system's robustness during its lifecycle.**
- **Regular Updates and Patch Management: Plan for regular updates, security patches, and bug fixes to address evolving threats and issues, ensuring the software remains secure, reliable, and compliant with changing standards and regulations over time.**

Identify critical issues that must be addressed by the architecture, such as: Are there hardware dependencies that should be isolated from the rest of the system? Does the system need to function efficiently under unusual conditions? Are there integrations with external systems? Is the system to be offered in different user-interfacing platforms (web, mobile devices, big screens,...)?

4 . Hardware Dependencies:

- **Isolation of Hardware Dependencies:** Isolate hardware-specific code and interactions (e.g., barcode scanners, receipt printers) to ensure that changes or failures in hardware components do not disrupt the overall system functionality. This isolation enhances the system's stability and flexibility.

5. Efficiency Under Unusual Conditions:

- **Robustness and Reliability:** Design the system to operate efficiently under unusual conditions, such as network outages or unexpected surges in customer transactions. Implement robust error handling, graceful degradation, and offline capabilities to ensure uninterrupted service even in adverse situations.

6. Integrations with External Systems:

- **Third-Party Integrations:** Identify and integrate with external systems, such as payment gateways and customer relationship management (CRM) tools. Ensure secure and seamless data exchange between the point of sale software and these external systems, maintaining data integrity and consistency.

7. Multi-Platform Support:

- **Cross-Platform Compatibility:** Design the user interface to be responsive and adaptable, supporting various platforms and devices, including web browsers, mobile devices (iOS and Android), and large screens (for display in-store analytics or advertisements).
- **Consistent User Experience:** Ensure a consistent user experience across different platforms, maintaining uniformity in design, features, and performance.

8. Data Security and Compliance:

- **Regulatory Compliance:** Ensure compliance with industry standards and regulations (e.g., PCI DSS for payment data) to protect customer data and avoid legal complications.
- **Login with auth:** Validate the type of user of the session and his permissions.

9. Scalability and Performance:

- **Scalability Planning:** Architect the system to be scalable, accommodating growth in the number of users, transactions, and data volume. Employ load balancing and distributed processing techniques to handle increased demand efficiently.
- **Performance Optimization:** Optimize database queries, use caching mechanisms, and employ efficient algorithms to minimize response times and maximize system performance, even during peak usage periods.

10. Maintenance and Upgrades:

- **Modular Architecture:** Adopt a modular architecture that allows for seamless updates and maintenance of individual components without disrupting the entire system. This modularity facilitates easier bug fixes, feature enhancements, and technology upgrades.
- **Automated Testing:** Implement automated testing procedures to detect issues early, ensuring that new updates or features do not introduce unforeseen problems
- into the system.

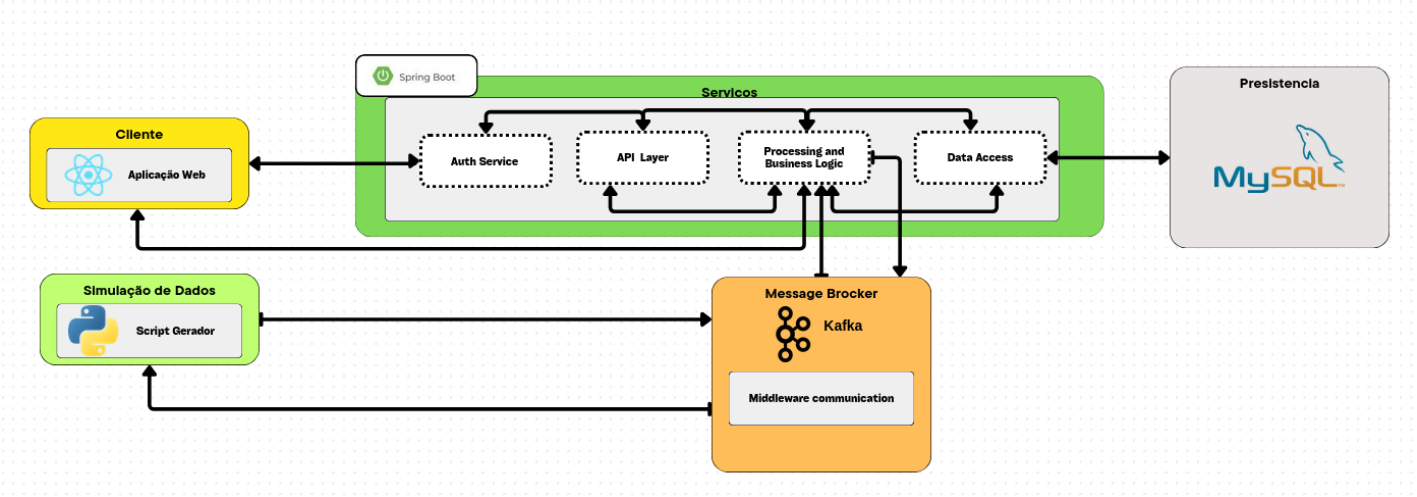
The project structure will follow the principles of this set of architectures :

- **1 - Data generation and Middleware data queues :** A simulated sensor that simulates sales performed by a certain employee identified by his id and products (these data come from the api)

- **2 - Database layer:** A single Architecture to store data for and return data through message queues for the server-side application mysql (which was chosen by its simplicity, stability and wide community for features support)

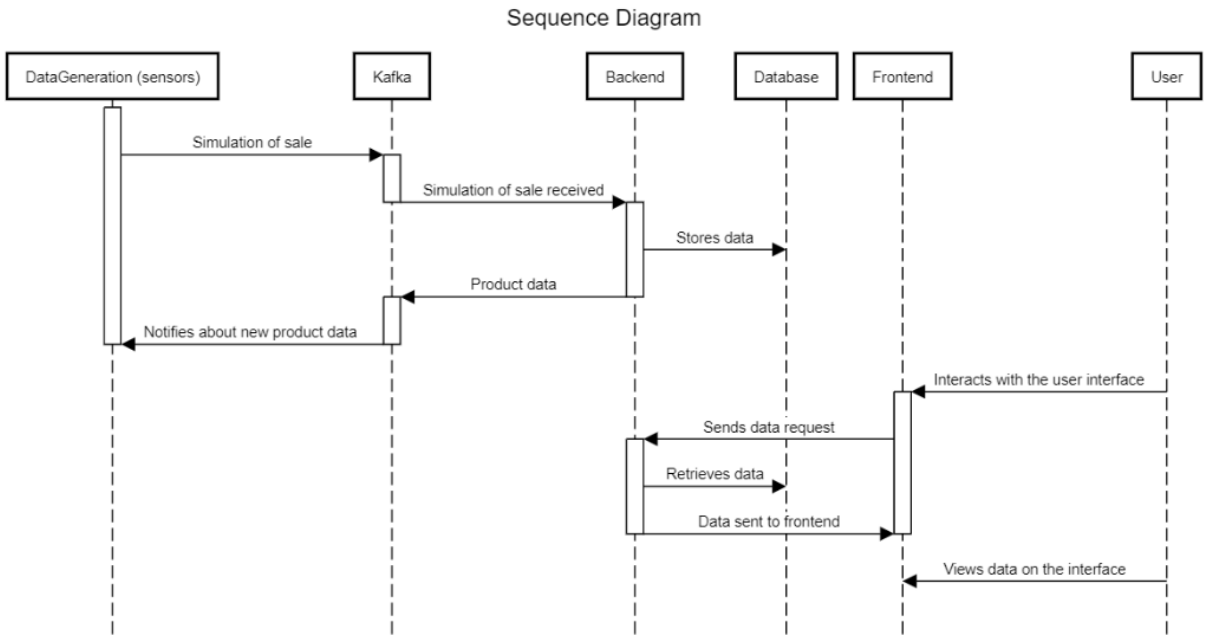
- **3 - API layer :** Multi-layered Architecture service that, in a scope of a single service/application, will handle the communication of the server application with the client-side application, bizz logic, data access and any other type of logical configuration of the project(such as external services or security handle).

- **4 - Client layer :** A Progressive Web Application (PWA) for implemented as a user web Interface developed with react library, which is a pretty popular, widely known (with tons of plugins and other tools for development) and powerful lib to build single-page dynamic web app, which is the case for this project. This service for itself will provide the login (authenticated) for the user and provide visual access to the data of the stock, fetching from the **API layer.**


Spring Boot
Servicos
Cliente
Aplicação Web
Auth Service
API Layer
Processing and Business Logic
Data Access
Presistencia
MySQL
Simulação de Dados
Script Gerador
Message Brocker
Kafka
Middleware communication

Module interactions

→ explain how the identified modules will interact. Use sequence diagrams to clarify the interactions along time, when needed



The data generation layer (sensors) simulates sales by sending information about transactions through Kafka. Then, Kafka receives and relays the sales simulation to the Backend, which stores the sales-related data in the database. Additionally, the Backend notifies Kafka about the new product data. The data generation layer (sensors) receives this notification, completing the cycle.

Next, the Frontend engages with the user, initiating a data request to the Backend. The Backend responds by retrieving the requested data from the Database and transmitting it back to the Frontend. The Frontend, in turn, communicates with the Backend to fetch and display the data on the user interface.

It is crucial to emphasize that this database storage process includes data validation and integrity checks at the Backend layer.

The data generated on the data gen layer must be synchronized in order to present the most up to date (real time simulated) to the client side (front-end), which is the responsibility of the back-end.

- **Data Synchronization Strategy**:

In our system architecture, a robust data synchronization strategy has to be imperative to maintain consistency and coherence across the distributed components. Here's a comprehensive strategy about how are we planning to approach it:

-Message-Driven Middleware (Kafka): we utilize Kafka as the message-driven middleware to facilitate communication between data-producing components (DataGeneration) and data-consuming components (Backend). Kafka acts as a reliable, scalable, and fault-tolerant backbone for data synchronization.

-Monitoring and Logging: we will implement extensive monitoring and logging specifically for synchronization processes. Detailed logs capturing synchronized data, detected conflicts, and the resolution outcomes will be maintained. Also, we can utilize centralized logging systems for streamlined analysis and to provide a comprehensive view of synchronization activities.

-Data Validation Checks: it is important to incorporate data validation checks to ensure the integrity of synchronized data: implement mechanisms to verify that synchronized data adheres to predefined constraints, preventing the propagation of erroneous information, Log validation results are also crucial, highlighting any discrepancies or issues identified during the synchronization process.

-Logging Server-Side Details: Enhance logging capabilities on the server side, capturing relevant details about data generation (DataGeneration) and database interactions. Log information related to the origin of synchronized data, timestamps, and any anomalies detected during synchronization. Centralize server-side logs for comprehensive auditing and analysis, aiding in troubleshooting and performance optimization.

By integrating these elements into our data synchronization strategy, we aim to ensure a consistent and reliable flow of data across our distributed system. The emphasis on logging, particularly on the server side, provides valuable insights into the synchronization process, making it easier to trace the source of data, monitor for conflicts, and maintain overall data integrity. This approach aligns with our system's architecture and supports the seamless interaction of components.

Applying these mechanisms to our 6 epics:

Epic 1: Inventory Management:

·Event Triggering: Notify Alex when a new product is successfully added to the inventory or when an existing product's information is updated.

·Backend Notification Service: Develop a backend notification service to handle events related to inventory management, ensuring Alex receives timely notifications.

Epic 2: Up-to-date Product Information

·Event Triggering: Send notifications to Sarah when product information, including pricing adjustments based on stock market variations, is updated.

·Backend Notification Service: Implement a backend service that identifies relevant events triggering pricing updates and sends notifications to Sarah.

Epic 3: Streamlined Checkout Process

·Event Triggering: Notify Sarah when a customer successfully completes the checkout process, indicating a successful transaction.

·Backend Notification Service: Create a backend service to handle checkout-related events and send notifications to Sarah for successful transactions.

Epic 4: Inventory Supervision

·Event Triggering: Send automatic alerts to Alex when popular products reach a defined low-stock threshold.

·Backend Notification Service: Develop a backend service to monitor inventory levels and trigger notifications for low-stock situations.

Epic 5: Sales Analytics Dashboard

·Event Triggering: Notify Alex when new sales data is available for the analytics dashboard.

·Backend Notification Service: Implement a backend service to process and analyze sales data, triggering notifications for Alex when the dashboard is updated.

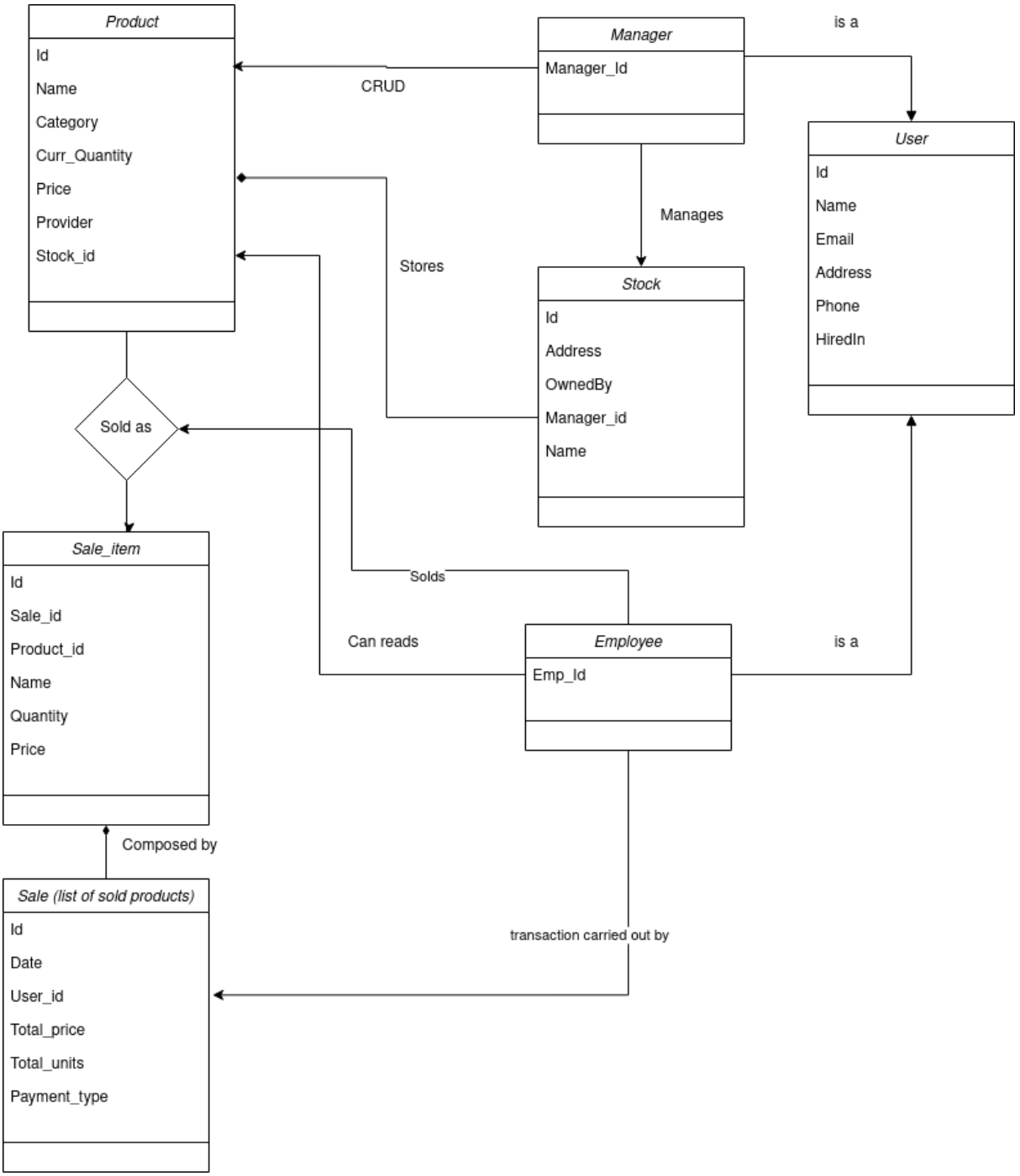
Epic 6: Employee Performance Tracking

·Event Triggering: Send notifications to Alex when new employee performance reports are generated.

·Backend Notification Service: Develop a backend service to track cashier performance, generating notifications for Alex based on report updates.

By aligning push notification mechanisms with specific events in each epic, we ensure that relevant stakeholders receive timely and personalized updates related to their areas of responsibility and interest within the system.

4 Information perspetive



5 References and resources

<document the key components (e.g.: libraries, web services) or key references (e.g.: blog post) used that were really helpful and certainly would help other students pursuing a similar work>

- https://www.xenonstack.com/insights/service-oriented-architecture-vs-microservices
    - https://www.ibm.com/topics/microservices
    - https://spring.io/projects/spring-boot/
    - https://react.dev/
    - https://medium.com/trendyol-tech/event-driven-microservice-architecture-91f80ceaa21e
    - https://spring.io/projects/spring-security/
    - https://medium.com/javarevisited/10-rest-api-best-practices-cd12e3904d00
    -