POS System for Small/Medium Restaurant Operations implemented with C++

Sánchez, K[1 [0009−0003−0311−9170]] and Pineda, I[2,3 [1111−2222−3333−4444]]

Yachay Tech University, Ibarra, Ecuador

Abstract. This work examines the operational challenges faced by small and medium-sized quick-service restaurants when attempting to scale their manual sales, inventory management, and daily accounting processes. Using a case study of an establishment in Urcuquí, Ecuador, a flexible and robust point-of-sale (POS) system is presented, specifically designed to modernize and optimize restaurant workflows. The proposed solution includes an intuitive terminal-based interface, real-time order processing, automated daily closing reports, and dynamic data export capabilities in JSON and CSV formats for further analysis and integration. Developed entirely in C++, the system leverages object-oriented programming, STL containers and algorithms, template-based abstractions, and concurrent data structures to ensure efficiency, scalability, and maintainability in demanding operational environments.

Keywords: Small and medium-sized enterprises (SMEs) · Real-time monitoring · Interface · Data structure

1 Introduction

The fast-food industry is at the crossroads of tradition and innovation, undergoing a digital revolution that is transforming the way businesses operate at all levels. As companies expand, the operational landscape becomes increasingly complex: managing sales, inventory, and daily accounting is no longer a simple matter of record-keeping, but a complete mix of data, human resources, and customer expectations. In this changing context, the limitations of manual processes become apparent. What was once sufficient for a small establishment quickly becomes a source of inefficiency and error as transaction volume increases and menu offerings diversify. Handwritten ledgers, spot inventory checks, and end-of-day reconciliations not only consume valuable time but also create opportunities for errors and data loss. [1]

The push toward digitalization is especially pronounced among small and medium-sized businesses (SMEs), which often lack the resources for large-scale enterprise solutions but face the same pressures to modernize as their larger counterparts. For these businesses, adopting point-of-sale (POS) systems represents a strategic turning point. Far from being simple digital cash registers, modern POS platforms serve as the hub of restaurant operations, integrating order management, sales tracking, inventory control, and even historical sales

management. Shin2013 By automating routine tasks and centralizing data, POS systems enable real-time visibility into sales trends, inventory levels, and staff performance, allowing managers to make informed decisions and respond proactively to changing business conditions.

The operational benefits of POS adoption are numerous. Automated inventory management reduces waste from excess inventory and mitigates the risk of lost sales due to stockouts. Digital records simplify reporting for fiscal and business analysis. Perhaps most importantly, the data generated by these systems forms the basis of advanced analytics, enabling time series forecasting, demand prediction, and strategic planning. [3] In an industry where margins are often razor-thin and customer preferences change rapidly, the ability to leverage data for competitive advantage is no longer optional: it is essential.

This paper presents the design and implementation of a custom POS solution tailored to the specific needs of a quick-serve restaurant in Urcuquí, Ecuador. The system was built from the ground up using contemporary C++ programming paradigms, including object-oriented design, STL containers, and template abstractions, ensuring both performance and extensibility. Attention is paid to the modularity of the codebase, allowing for seamless integration of new features and adaptation to changing operational requirements. The backend leverages robust data structures for order and inventory management, while the frontend provides an intuitive terminal interface.

Beyond the core functionalities of order processing and sales tracking, the system is designed with interoperability in mind. Data is exported in JSON format, facilitating integration with Python-based analytics tools and enabling CSV report generation for other business intelligence applications. The complete system architecture, including source code and documentation, is available at: https://github.com/KevJoss/FastFood-POS-System.git. This project seeks not only to address the immediate operational challenges faced by small and medium-sized restaurants but also to lay the foundation for a scalable, data-driven approach to restaurant management that can evolve alongside the industry's ongoing digital transformation.

1.1 Problem Statement

As noted above, the case study concerns a local fast-food establishment located in Urcuquí, Ecuador, called Gustav's. Consequently, it is helpful to explore the contextual framework surrounding the main problem faced by this establishment; while the nature of such problems may show slight variations depending on the restaurant, their prevalence tends to increase in line with the rapid expansion within this sector.

Gustav's has been in operation for approximately fourteen years. The restaurant is especially known for its french fries and other dishes: - Such as "papipollos", "salchipapas", hamburgers, wings and various potato-based dishes. - Complements such as chaulafan, juices, sodas and milkshakes. Throughout its operating history, this restaurant has experienced growth, which has required

expansions in terms of physical infrastructure, menu diversification, kitchen capacities and personnel.

In addition to these modifications, the increasing daily influx of customers has made the conventional methodology of manual sales recording increasingly inadequate to meet operational requirements. For the purposes of sales documentation and the subsequent digitization process, the restaurant currently employs a rudimentary system, which fails to accommodate current needs.

This manual system presents a multitude of challenges, especially when multiple employees use the same notebook throughout the week. Problems include confusing handwriting, missing entries, and physical degradation of the document due to exposure to culinary elements such as cooking oils.

Inefficiencies are further compounded during end-of-day procedures, where all register entries must be manually transcribed into Ecuador's electronic invoicing systems, a labor-intensive task that requires individual entry of each transaction and imposes considerable burden and wear and tear on the person responsible.

Another complication is inventory management and analysis of daily, weekly, monthly and annual sales. Currently, inventory replenishment decisions are executed without prior analysis of historical sales data, culminating in suboptimal buying patterns. This therefore results in inventory spoilage or stock-outs.

These challenges demonstrate the need for an integrated digital system capable of establishing an infrastructure for order automation, a robust database that facilitates comprehensive data analysis and enables time series forecasting to improve purchasing decisions. Such a system, designed in C++ with the ability to integrate Python for analytical purposes, would integrate seamlessly with POS operations to address customer needs.

1.2 Proposed Solution

Implementing a comprehensive point-of-sale (POS) system represents a good strategy for solving the operational challenges mentioned in the previous section. This integrated digital system will enable real-time order processing, automated data retention, and systematic analytical capabilities, thus improving efficiency. The proposed system architecture comprises several essential functional components, which are developed in the following subsections.

- **A. Order Management/Order Entry Interface:** An intuitive terminal interface displays menu items organized by category (main dishes, combos, drinks, add-ons). Service personnel select products and record orders.
- **B. Order Configuration:** The system requests the specification of the service type (serve-in or takeout).

The flow that this project will follow is shown in the following Figure 1

Fig. 1. System architecture and file interaction diagram.

Each confirmed order is assigned a unique alphanumeric identifier for subsequent tracking and modification.

2 Methodology

This section outlines the workflow and conceptual underpinnings that guided the development of the POS system. The starting point was the construction of a general menu, hosted in the folder `environment`, composed of the files `environment.cpp` and `environment.hpp`. These modules manage the main interactive interface, structured in five sections: Order List, Statistics, Products, Make Sale and End Day. Each section responds to a specific operational need and will be broken down in later sections.

2.1 Environment and interaction management

The user interaction logic is based on the capture of numeric inputs from the keyboard, using useful functions for visual display, such as text centralization and

generation of console splits. The `<thread>` library was incorporated along with `<chrono>` to simulate delays and smooth transitions between sections, improving the user experience. In addition, a screen-clearing mechanism (*clear console*) was implemented to keep the interface tidy and readable after each operation.

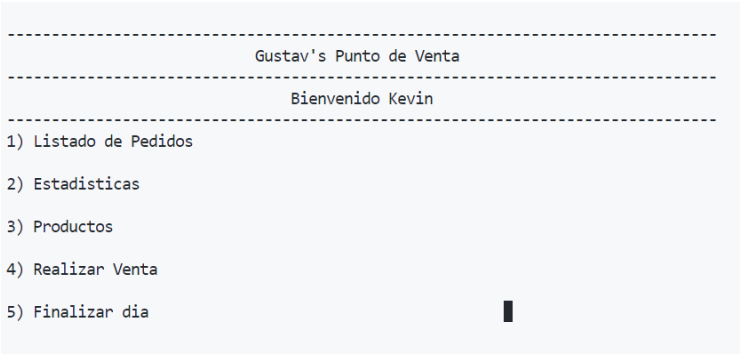

Fig. 2. Interactive menu interface

2.2 Product management and modeling

The first technical challenge was to obtain and normalize the restaurant's product database. The initial input was an Excel file, converted to CSV and then cleaned and transformed to JSON format using Python scripts. This transformation allowed segmenting the products into categories -dishes, combos, wings, extras and drinks- (The following figures (figure 5 and figure 4) show the visual representations of JSON files for dishes and wings.) and defining a flexible data structure, where each entry contains attributes such as name, price, and in some cases, takeaway price, flavor or volume.

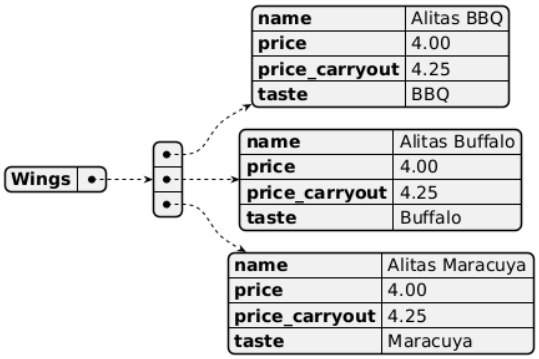

Fig. 3. JSON file for Wings products.

The use of JSON as an intermediate format responds to the need for a dynamic database, capable of being modified without direct intervention in the source code, thus facilitating scalability and maintenance.

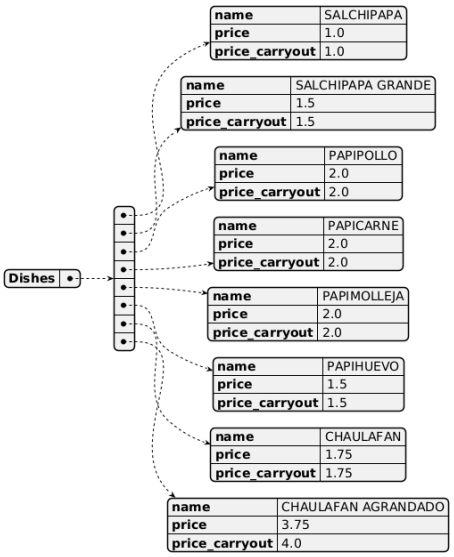

Fig. 4. JSON file for Dishes products.

2.3 Object-oriented design and polymorphism

The class architecture, defined in `MenuItems.cpp` and `MenuItems.hpp`, is based on object-oriented design principles. A hierarchy was established where the abstract class `Product` encapsulates the essential attributes and behaviors, and from which specializations such as `CarryoutProduct`, `Dish`, `Wings`, `Extra` and `Beverage` derive. This approach allows modeling the heterogeneity of menu products, taking advantage of polymorphism to manipulate them in a uniform way using smart pointers (`std::shared_ptr<Product>`). Dynamic memory management through these pointers ensures efficiency and safety, avoiding leaks and facilitating shared object reference among multiple active orders.

2.4 Dynamic loading and efficient product search

System initialization involves loading products from JSON files, using the `nlohmann/json` library for parsing and insertion into STL containers. This separation between business logic and data persistence allows the menu to be updated in an agile and secure manner. Search algorithms by ID and name were developed, optimizing product selection during order taking and minimizing the possibility of operational errors.

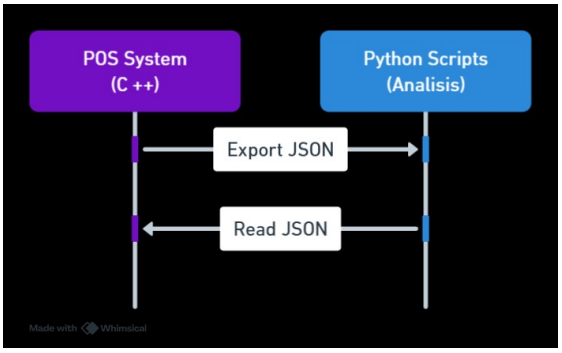

Fig. 5. Pipeline of JSON files.

2.5 Order management and persistence

The sales flow is structured as an iterative interaction: the user selects the type of order (to take away or consume locally) and adds products using their identifiers and quantities. Each order is stored in a polymorphic structure, where products are represented as smart pointers to the base class together with the requested quantity. This design allows the same product to be referenced in several orders simultaneously, optimizing resources and simplifying the elimination logic after delivery.

For order persistence, an export function was implemented that serializes the general order map (`GeneralMap`) to a JSON file at the end of the day. This file stores, for each order, its identifier, modality (carryout or local consumption), and the list of products with quantities and final prices. The structured format facilitates interoperability with other tools and systems.

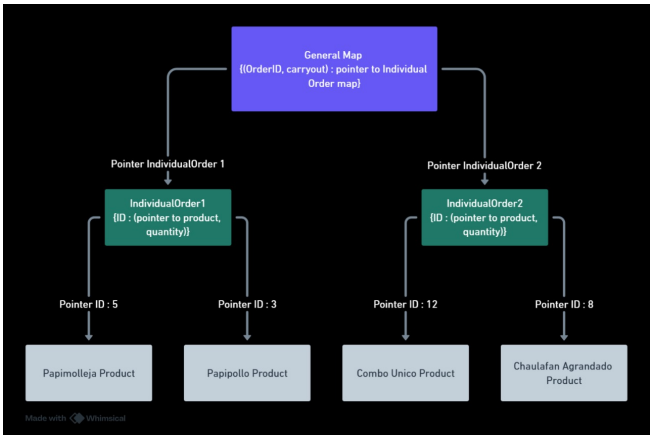

Fig. 6. Structure workflow of data

2.6 Data analysis in Python

The final stage of the methodological flow consists in the exploitation of the generated data (Figure 11). The JSON file of orders is imported into Python, where, by means of the libraries `json` and `pandas`, it is transformed into a DataFrame.

	Pedido ID	Para llevar	Producto	Cantidad	Precio
0	1	False	PAPIPOLLO	5	10.00
1	10	True	HAMBURGUESA	1	3.50
2	2	False	SALCHIPAPA	3	3.00
3	2	False	HAMBURGUESA	2	7.00
4	2	False	Alitas BBQ	1	4.00
5	2	False	CocaCola 1.3l	1	1.75
6	3	True	COMBO 1	2	7.00
7	4	False	PAPIPOLLO	2	4.00
8	4	False	CHAULAFAN	1	1.75
9	4	False	Sodas	2	1.00
10	5	False	SALCHIPAPA GRANDE	3	4.50
11	5	False	PAPIMOLLEJA	2	4.00
12	6	False	PAPIPOLLO	4	8.00
13	6	False	COMBO DOBLE	1	3.75
14	6	False	Alitas Maracuya	1	4.00
15	6	False	Sodas 1.3 litros	1	1.50
16	7	True	COMBO 2	1	3.50
17	7	True	Sodas	1	0.50

Fig. 7. DataFrame from JSON file

This tabular structure allows statistical analysis, filtering and visualization, providing valuable information for strategic decision making. The integration between C++ and Python demonstrates the importance of designing open systems, capable of interacting with different languages and platforms according to analytical or business needs.

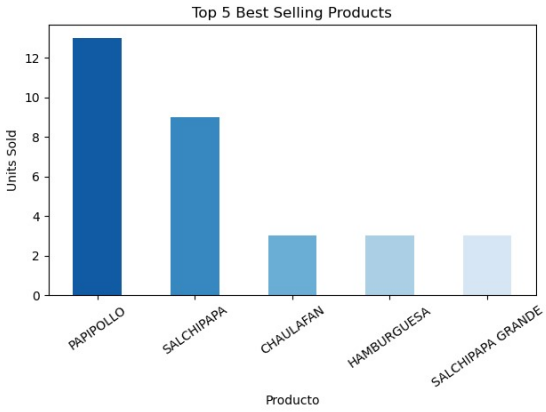

Fig. 8. Top 5 best selling products.

In summary, the implemented methodology articulates concepts of object-oriented design, efficient memory management, decoupling between logic and data, and interdisciplinary analytical exploitation. This approach not only ensures the robustness and adaptability of the system, but also lays the groundwork for future extensions and enhancements to the restaurant's operating environment.

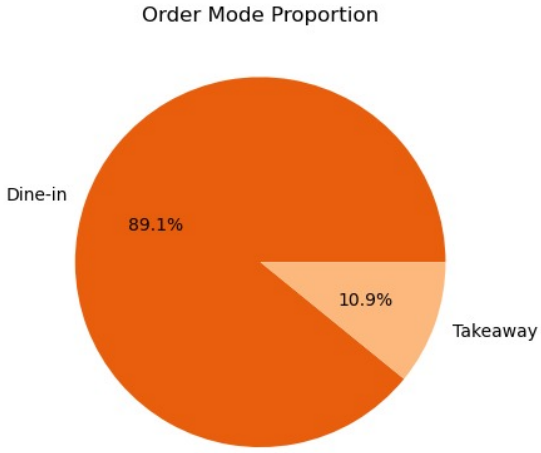

Fig. 9. Proportion of order modes (dine-in vs takeaway).

Finally, a summary graph of the interconnection between all the project files and the database is presented.

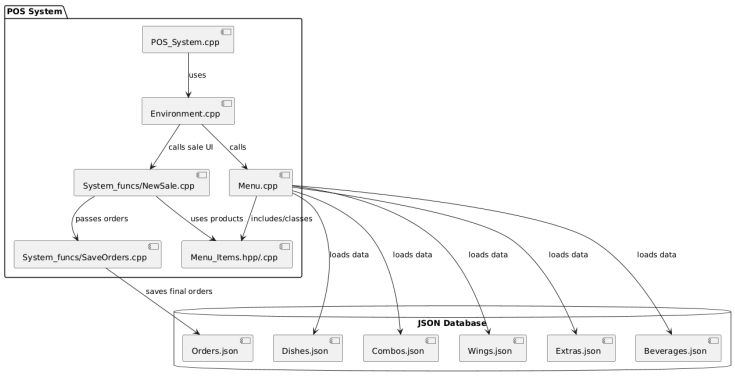

Fig. 10. Files and database conections

3 Conclusions

The development and implementation of the POS system described in this work demonstrates the tangible benefits of digital transformation for small and medium-sized restaurant operations. By leveraging object-oriented programming in C++ and integrating dynamic data management through JSON and Python, the system achieves a robust, modular, and extensible architecture. The solution not only streamlines order processing and daily reporting, but also lays the foundation for advanced data analysis and informed decision-making. The modular design, with clear separation between business logic and data persistence, ensures maintainability and scalability, allowing the system to adapt to evolving operational requirements. The successful integration of C++ and Python further highlights the value of interoperability in modern software solutions, enabling seamless transitions between real-time operations and post-hoc data analysis.

4 Recommendations

For establishments considering the adoption of similar POS solutions, it is advisable to:

- Maintain a clear separation between user interface, business logic, and data storage to facilitate future updates and debugging.
- Regularly back up both the product database and order records to prevent data loss and ensure business continuity.
- Provide basic training for staff on the use of the system, emphasizing the importance of accurate data entry and regular system maintenance.

- Periodically review and update the product database to reflect menu changes, pricing adjustments, and new offerings.
- Consider integrating additional security measures, such as user authentication and access controls, especially as the system scales.

5 Future Work

While the current system addresses the core needs of order management and data analysis, several enhancements are envisioned to further increase its utility and user-friendliness:

- **Order Editing and Deletion:** Implement functions that allow users to modify or remove existing orders, providing greater flexibility and error correction during busy service periods.
- **Payment Attributes:** Extend the system to record payment methods (e.g., cash, bank transfer), enabling more detailed financial tracking and reporting.
- **Payment Interface and Change Calculation:** Develop an interface for entering payment amounts and automatically calculating the change due, streamlining the checkout process and reducing the risk of human error.
- **User Roles and Permissions:** Introduce differentiated access levels for administrators, cashiers, and kitchen staff to enhance security and workflow management.
- **Integration with Inventory Management:** Link sales data with inventory tracking to enable real-time stock updates and automated restocking alerts.
- **Graphical User Interface (GUI):** Explore the development of a graphical interface to further improve usability, especially for users less familiar with terminal-based systems.

These future developments will not only enhance the operational efficiency of the POS system but also contribute to a more comprehensive digital ecosystem for restaurant management.

6 Appendix

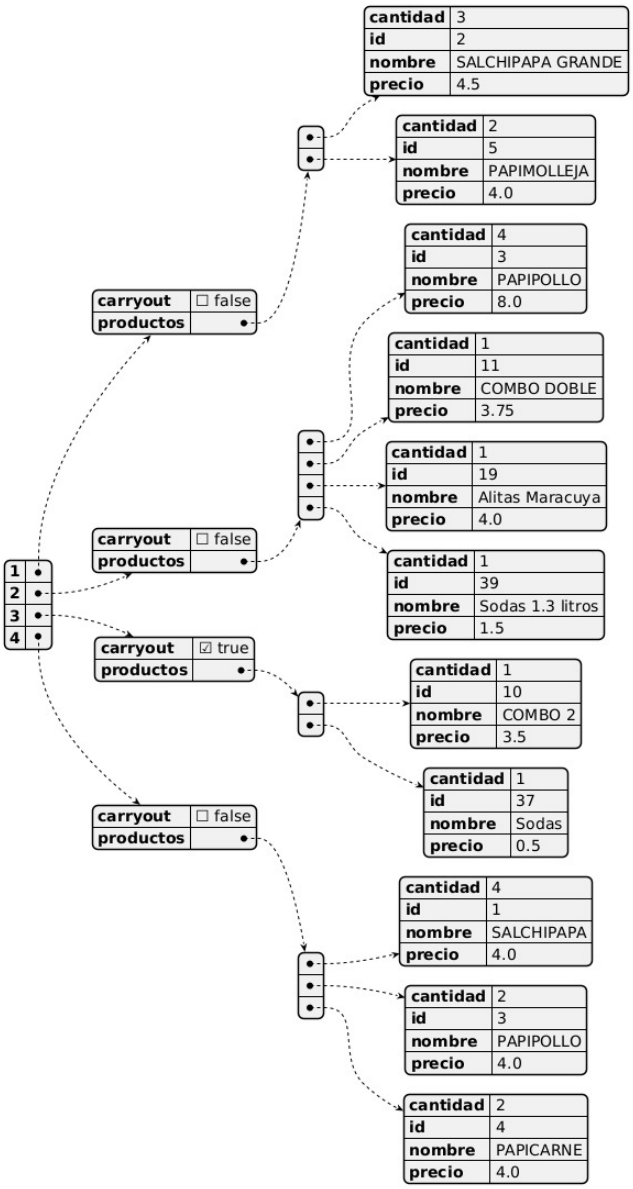

Fig. 11. Exported JSON file

```
--------------------------------------------------------------------------------
                            Listado de pedidos
--------------------------------------------------------------------------------

Pedido ID: 1 | Para llevar: No
  Cantidad: 3 | Nombre: SALCHIPAPA GRANDE | Precio: 4.5 $
  Cantidad: 2 | Nombre: PAPIMOLLEJA | Precio: 4 $
Total del pedido: 8.5 $

--------------------------------------------------------------------------------
Pedido ID: 2 | Para llevar: No
  Cantidad: 4 | Nombre: PAPIPOLLO | Precio: 8 $
  Cantidad: 1 | Nombre: COMBO DOBLE | Precio: 3.75 $
  Cantidad: 1 | Nombre: Alitas Maracuya | Precio: 4 $
  Cantidad: 1 | Nombre: Sodas 1.3 litros | Precio: 1.5 $
Total del pedido: 17.25 $

--------------------------------------------------------------------------------
Pedido ID: 3 | Para llevar: Si
  Cantidad: 1 | Nombre: COMBO 2 | Precio: 3.5 $
  Cantidad: 1 | Nombre: Sodas | Precio: 0.5 $
Total del pedido: 4 $

--------------------------------------------------------------------------------
Pedido ID: 4 | Para llevar: No
  Cantidad: 4 | Nombre: SALCHIPAPA | Precio: 4 $
  Cantidad: 2 | Nombre: PAPIPOLLO | Precio: 4 $
  Cantidad: 2 | Nombre: PAPICARNE | Precio: 4 $
Total del pedido: 12 $

--------------------------------------------------------------------------------
```

Fig. 12. Output of order summary to terminal.

```
--------------------------------------------------------------------------------
                                  Productos
--------------------------------------------------------------------------------
                              Platos Individuales
--------------------------------------------------------------------------------

 1) SALCHIPAPA           2) SALCHIPAPA GRANDE    3) PAPIPOLLO
 4) PAPICARNE            5) PAPIMOLLEJA          6) PAPIHUEVO
 7) CHAULAFAN            8) CHAULAFAN AGRANDADO
----
                                  Combos
--------------------------------------------------------------------------------

 9) COMBO 1             10) COMBO 2             11) COMBO DOBLE
12) COMBO UNICO         13) COMBO FAMILIAR      14) HAMBURGUESA
15) HAMBURGUESA DOBLE   16) HAMBURGUESA SOLA
17) Alitas BBQ          18) Alitas Buffalo      19) Alitas Maracuya

--------------------------------------------------------------------------------
                                  Extras
--------------------------------------------------------------------------------

20) ENSALADA            21) PORCION ARROZ       22) PORCION BROSTER
23) PORCION CARNE       24) PORCION HUEVO       25) PORCION MOLLEJA
26) PORCION PAPAS       27) PORCION SALCHICHA

--------------------------------------------------------------------------------
                                  Bebidas
--------------------------------------------------------------------------------

28) Jugo Mora           29) Jugo Guanabana      30) Batido Mora
31) Batido Guanabana    32) CocaCola 300ml      33) CocaCola 500ml
34) CocaCola 1lt        35) CocaCola 1.3l       36) Coca Cola Lata
37) Sodas               38) Sodas 500ml         39) Sodas 1.3 litros
40) Te 250ml            41) Te 550ml            42) Te 1 litro
43) Squiz 300ml         44) Pepsi 300ml         45) Pepsi 1 litro
46) Quintuple 300ml     47) Quintuple 1 litro   48) Tesalia Ice 500ml
49) Agua 600ml          50) Agua 1 litro        51) Gaseosa en vaso
```

Fig. 13. Output products into terminal

References

1. Söderbaum, P.: Environmental Economics: A Heterodox Approach. Routledge (2011)
2. Shin, Y.: The Impact of POS Systems on Fast-Food Business Operations. *Journal of Retail and Consumer Services*, **20**(2), 123–130 (2013). https://doi.org/10.1016/j.jretconser.2013.01.004
3. Dorsey, J., Wu, C., Zhang, L.: Leveraging Point-of-Sale Data for Improved Retail Decision Making. *International Journal of Information Management*, **37**(3), 188–197 (2017). https://doi.org/10.1016/j.ijinfomgt.2017.01.005
4. A. Smith and B. Jones, "Modern Point-of-Sale Systems for Small Restaurants", *Journal of Restaurant Technologies*, vol. 4, no. 2, pp. 123–145, June 2023.
5. S. Khalid, *Point of sale system* [Bachelor Thesis]. Govt Degree College, 2022.