Engenharia de Requisitos

Hugo Coelho - 1162086
Ilídio Magalhães - 1191577
Paulo Abreu - 1240481
Pedro Oliveira - 1240482

December 2024

1 Introduction

This project was developed as part of the 'Engenharia de Requisitos' course, within Group 3 of class M1A. The work focuses on applying software development best practices, emphasizing the use of Scrum as the agile methodology, story mapping for requirements prioritization, and clear role allocation within the team to ensure a structured and collaborative approach.

The objective was to develop a solution based on the initial requirements provided, revalidated with the client throughout the iteration process, while adhering to best practices in software architecture and development processes.

1.1 Roles

The roles within the team were defined as follows:

- **Product Owner:** Paulo Abreu was responsible for ensuring the prioritization of features and alignment with the client's needs.
- **Scrum Master:** Ilídio Magalhães oversaw the implementation of Scrum processes, facilitating team communication and ensuring adherence to the agile framework.
- **Team Developers:** Hugo Coelho and Pedro Oliveira contributed as team developers, focusing on implementing and delivering the features within the sprint goals.

During the final weeks, these roles rotated among all team members to ensure a balanced distribution of responsibilities and to provide everyone with a comprehensive understanding of each role's challenges and tasks.

2 User Stories

Stories

1. As a **Producer**, I want to define the catalog of products I produce, including name, brief description, photo, reference price, and delivery units for each product. These products can be simple (e.g., cured goat cheese) or composite (e.g., a basket of mixed vegetables).

2. As the **AMAP Manager**, I want to define a subscription period with a specified duration (e.g., semester), including start and end dates, as well as the delivery dates within the period, and automatically notify producers of the start of the process.

3. As a **Producer**, I want to create a product offering for the subscription period defined by AMAP management. I should be able to choose delivery dates, select products from my catalog for delivery on specific dates, and define payment methods and their breakdown (e.g., full payment or installments).

4. As a **Co-Producer**, I want to subscribe to the product offering created by producers, selecting the products and quantities I want to receive on each date and choosing the payment method and breakdown based on available options.

5. As the **AMAP Manager**, I want to trigger the calculation of amounts to be paid by co-producers and amounts to be received by producers based on subscriptions and payments made.

6. As a **Producer**, I want to deliver the subscribed products, being able to adjust the delivered quantities relative to the subscription to accommodate production changes (positive or negative).

7. As a **Manager**, I want to consult critical KPIs for management, such as the value delivered by producer per delivery and per period, and the average value subscribed by co-producer per delivery and per period.

8. As the **AMAP Manager**, I want to trigger the calculation of differences between amounts paid and amounts for deliveries, adding these values to the co-producers' account balances.

3 Story Mapping

3.1 User Story 1: Producer - Catalog Definition

User Story 1

MVP:
Focuses on defining product catalog details, including name, description, photo, reference price, delivery units, and product type (Simple or Basket). It also includes listing all products and displaying their basic details.

Justification:
These features are essential for creating and managing a product catalog, providing users with a functional system for basic product setup and visibility.

MVI1:
Adds advanced features such as multiple photos per product, change history, dependency checks (e.g., ensuring products aren't linked to orders or baskets), and filtering products by type or price.

Justification:
Enhances the usability and reliability of the product catalog, improving user experience by offering better control and clarity over product management.

MVI2:
Introduces product categorization through tags, significant update notifications, product archiving for historical purposes, and sorting products by the date added.

Justification:
These features improve organization and communication, adding convenience and scalability for managing larger catalogs.

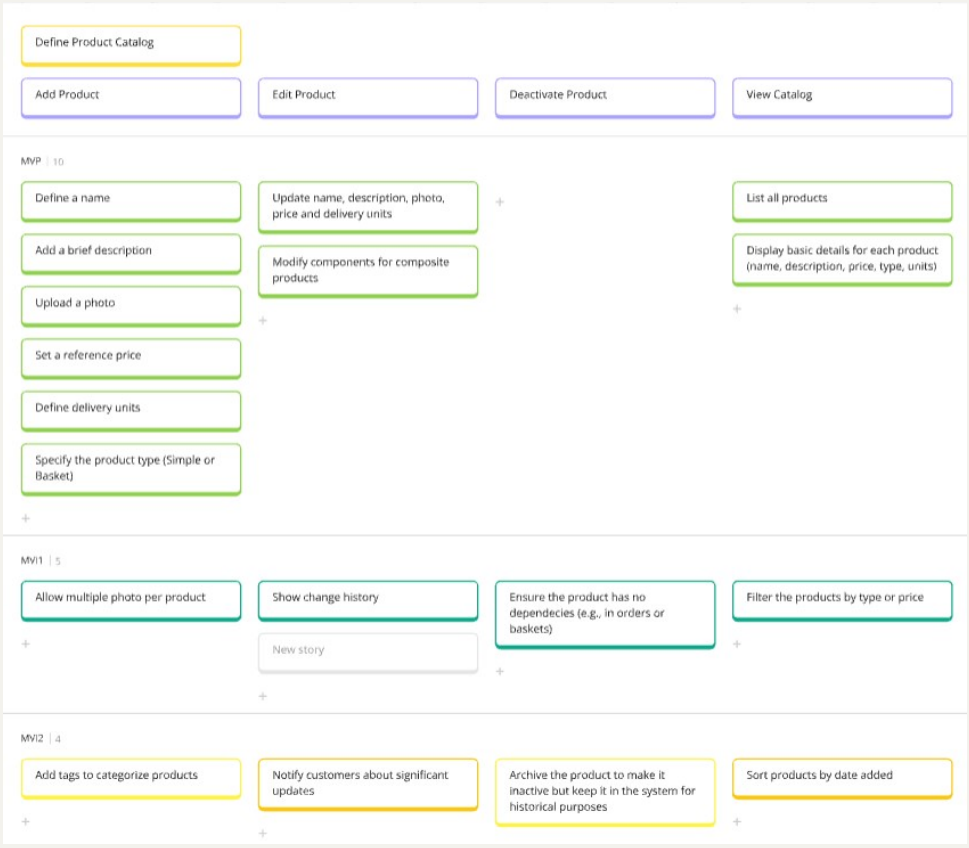

Figure 1: Producer - Catalog Definition

3.2 User Story 2: Subscription Period Management

User Story 2

MVP:
Features include defining a subscription period (name, start date, end date, list of delivery dates), editing these details, and notifying producers via email about the new or updated subscription period.

Justification:
These functionalities are crucial for enabling AMAP Administrators to set up and manage subscription periods effectively, ensuring the process is well-structured and communicated to producers.

MVI:
Adds validation for delivery dates (ensuring they fall within the subscription period) and subscription period names (ensuring no duplicates), as well as ensuring subscription periods have no unresolved dependencies.

Justification:
These improvements increase reliability and data integrity, preventing errors that could disrupt the subscription process or confuse users.

Backlog:
Includes sending email notifications to producers at the start of the subscription period.

Justification:
While helpful for communication, this feature is not critical for initial functionality and can be implemented later as an enhancement to producer engagement.

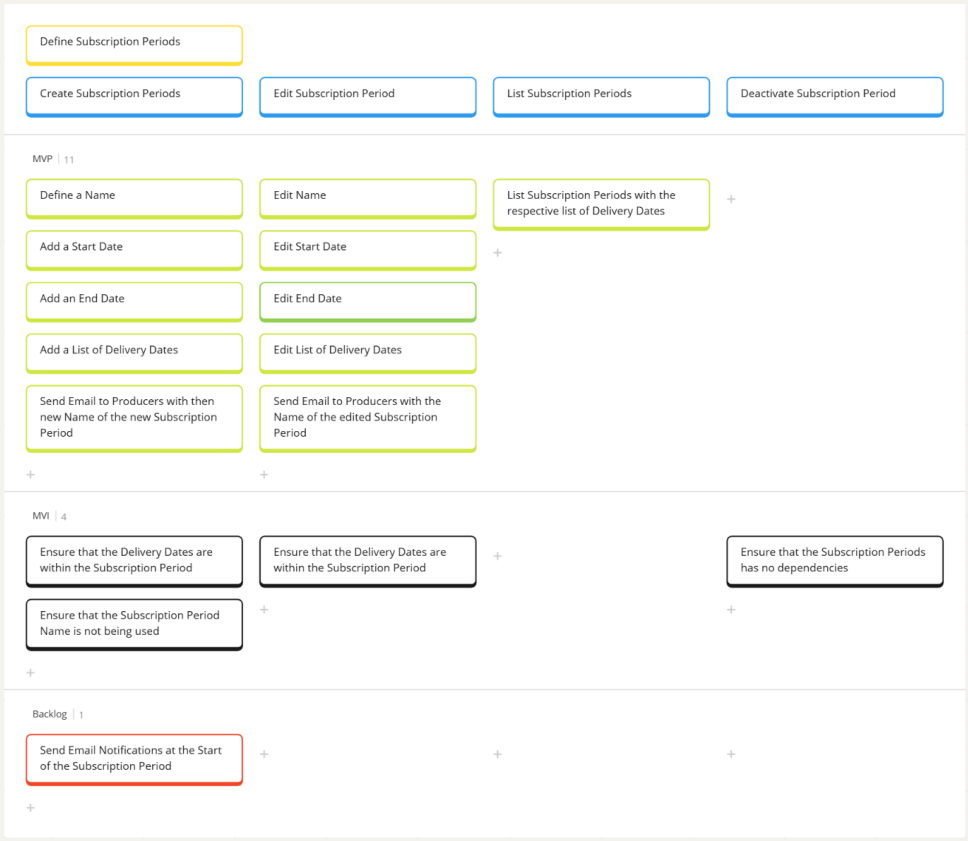

Figure 2: Subscription Period Management

3.3 User Story 3: Product Offer Management

User Story 3

MVP:
Includes choosing delivery dates, assigning products to delivery dates, and selecting payment methods and modes.

Justification:
These are essential for enabling users to configure and activate subscription services, ensuring the system's basic functionality is operational.

MVI1:
Adds validation of delivery dates, product availability display, and payment reminders.

Justification:
Improves accuracy and user trust by reducing errors, providing transparency, and ensuring timely payment follow-ups.

MVI2:
Introduces default delivery schedule suggestions and significant update notifications.

Justification:
Adds convenience for users by automating routine tasks and keeping them informed about changes.

Backlog:
Includes early delivery notifications and integration with automated payment systems.

Justification:
These are valuable but non-critical features, best suited for future updates to enhance the user experience.

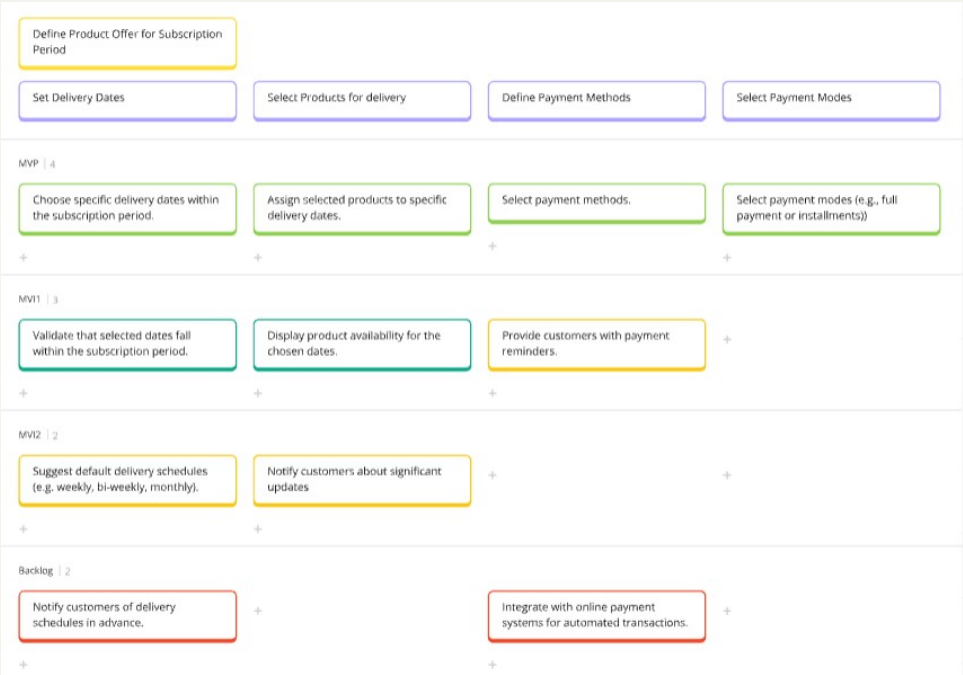

Figure 3: Product Offer Management

3.4 User Story 4: Select Product Offer

User Story 4

MVP:
Features include adding and editing delivery dates, product offering identifiers, subscription identifiers, product quantities, and subscription payments (including payment dates, amounts, and statuses). It also includes listing subscription periods with respective delivery dates.

Justification:
These are foundational for managing product offerings and subscriptions, ensuring users can define, modify, and track all relevant details necessary for the subscription process.

MVI:
Adds validations to ensure payment dates are not after delivery dates, subscription identifiers belong to co-producers, and the subscription period matches the product offering. Prevents co-producers from updating identifiers or modifying subscriptions after payments or cancellations. Also ensures no unresolved dependencies for the selected product offering.

Justification:
These features enhance reliability and data integrity, preventing errors and ensuring the system functions as expected, especially in critical scenarios involving payments and modifications.

Figure 4: Select Product Offer

3.5 User Story 5: Calculate Payments and Balances

User Story 5

MVP:
Features include adding co-producer names, producer names, subscription names, pending payments, and co-producer debt. Additionally, it includes calculating the amounts to be paid by co-producers and the amounts owed to producers.

Justification:
These features are foundational for tracking and calculating financial transactions between co-producers and producers, ensuring transparency and accurate payment management.

MVI:
Adds features such as pagination, filters, and the ability to export lists as reports.

Justification:
These enhancements improve usability by making the data easier to navigate, customize, and export for reporting or further analysis.

Backlog:
Includes sending notifications about financial transactions and balances.

Justification:
Notifications add value by keeping stakeholders informed, but they are non-critical for initial functionality and can be implemented later.

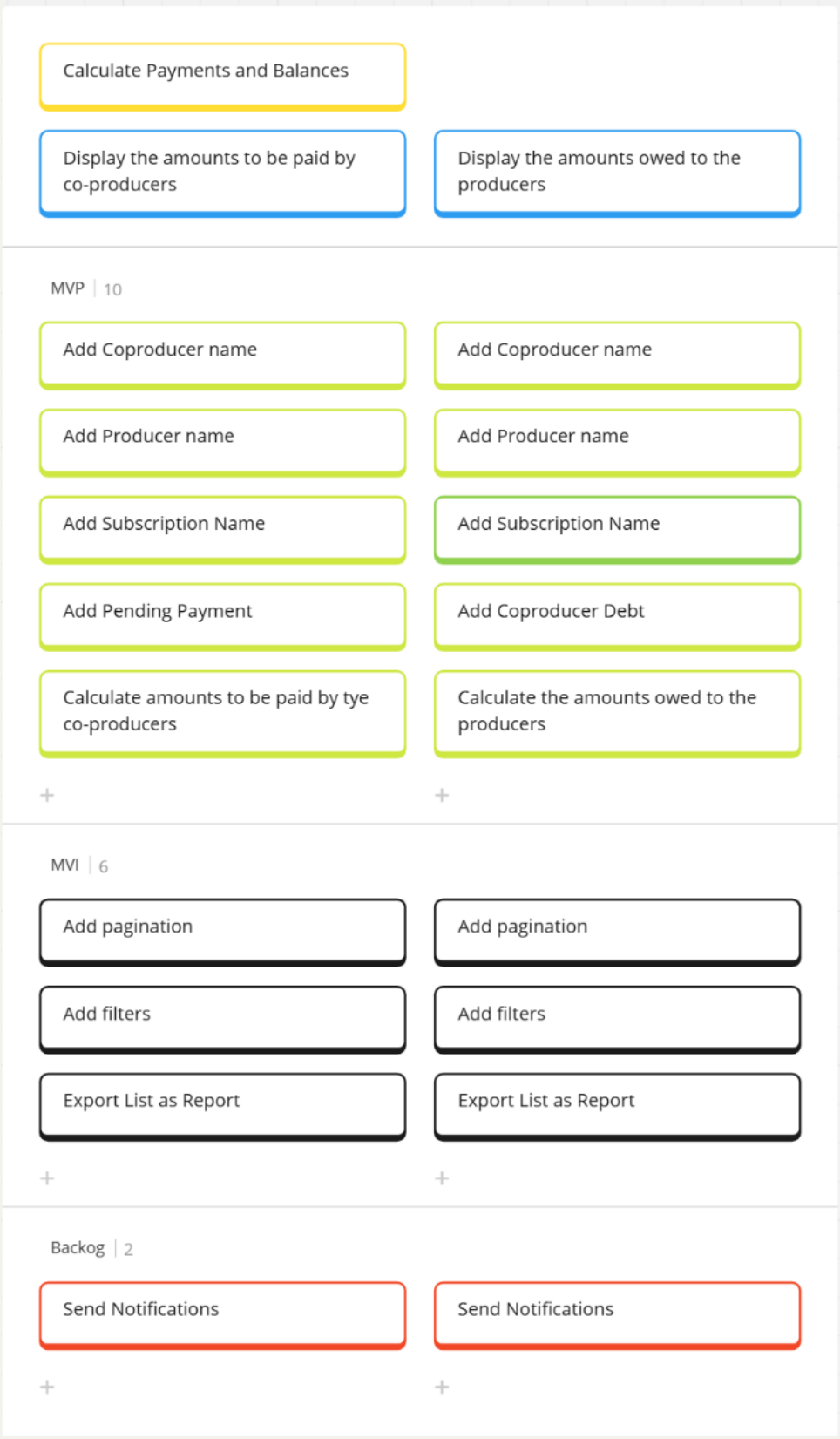

Figure 5: Select Product Offer

3.6 User Story 6: Update Quantity

User Story 6

MVP:
Features include displaying subscriptions, selecting product quantities, and displaying the total amount.

Justification:
These are foundational features required to view and interact with subscriptions, ensuring users can access and configure the necessary data.

MVI:
Adds features like allowing subscription selection, filtering by status/date, showing product and quantity details, sorting subscriptions, error notifications for invalid quantities, confirming payment, confirming product delivery, and marking delivery as complete.

Justification:
These enhancements improve the usability and reliability of the system by enabling detailed interactions with subscriptions, preventing errors, and supporting key transactional actions like payment and delivery confirmation.

MVI2:
Introduces advanced features such as a search bar with filters and reviewing payment details.

Justification:
These features enhance the user experience by providing efficient navigation and deeper insights into payments, which are valuable but not critical to the initial functionality.

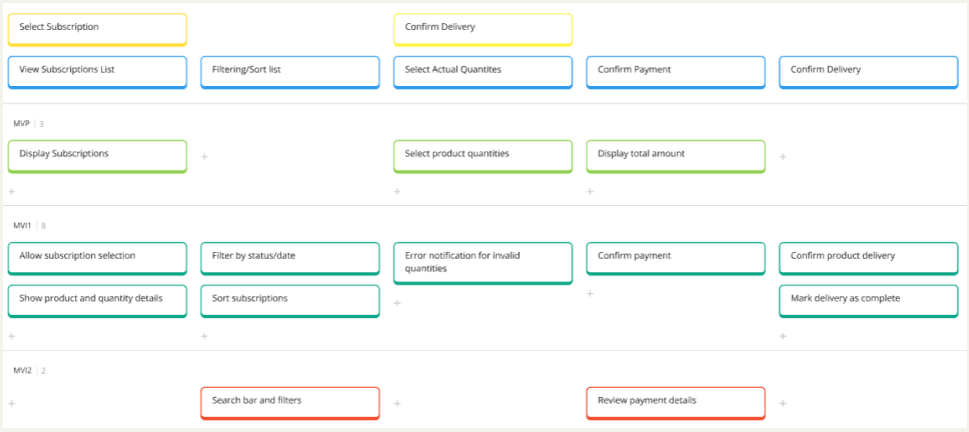

Figure 6: Update Quantity

3.7 User Story 7: Consult Critical KPIs for Management

User Story 7

MVP:
Features include displaying the average subscribed value for each delivery and the average subscribed value by co-producer.

Justification:
These are essential metrics that provide managers with baseline insights into delivery performance and co-producer participation.

MVI1:
Adds features such as filtering for specific producers, co-producers, or deliveries, selecting a specific subscription, comparing averages across periods, exporting KPI reports, creating a centralized dashboard, and displaying critical KPIs using graphs, tables, and aggregated metrics.

Justification:
These enhancements allow managers to analyze trends, focus on specific data points, and generate reports for more informed decision-making.

Backlog:
Includes allowing the manager to configure automatic reports.

Justification:
Automating reports improves convenience and efficiency but is not critical for initial functionality and can be added in later iterations.

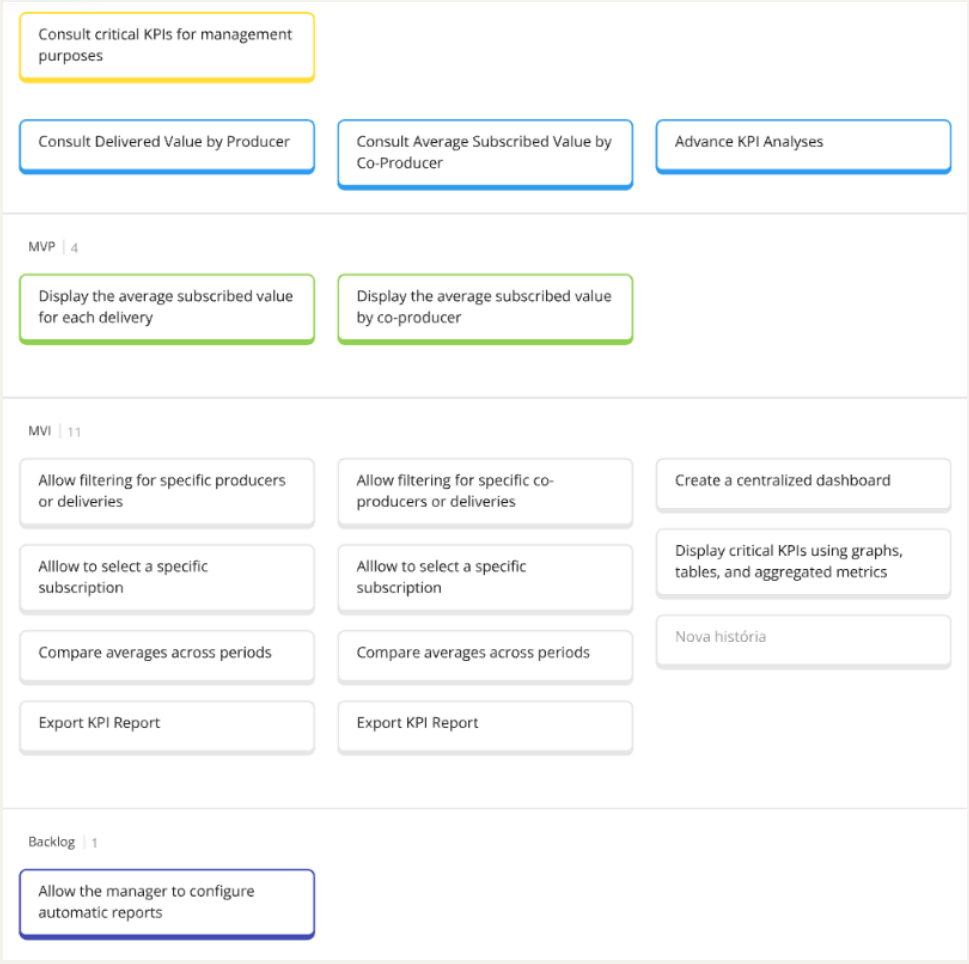

Figure 7: Consult Critical KPIs for Management

3.8 User Story 8: Trigger Calculation of Differences

User Story: Trigger Calculation of Differences

MVP:
Features include creating a button or UI element to trigger the calculation, processing the calculation, and displaying updated balances.

Justification:
These features are essential to initiate and complete the calculation process, ensuring users can view updated balances based on the triggered calculations.

MVI1:
Adds features like highlighting significant differences in the results to draw attention to critical changes or discrepancies.

Justification:
This enhancement improves usability and allows users to focus on important information, aiding decision-making and resolving potential issues more effectively.

Backlog:
Includes exporting the results.

Justification:
This feature provides added functionality for generating reports or storing data but is not critical for the initial calculation and display functionality.

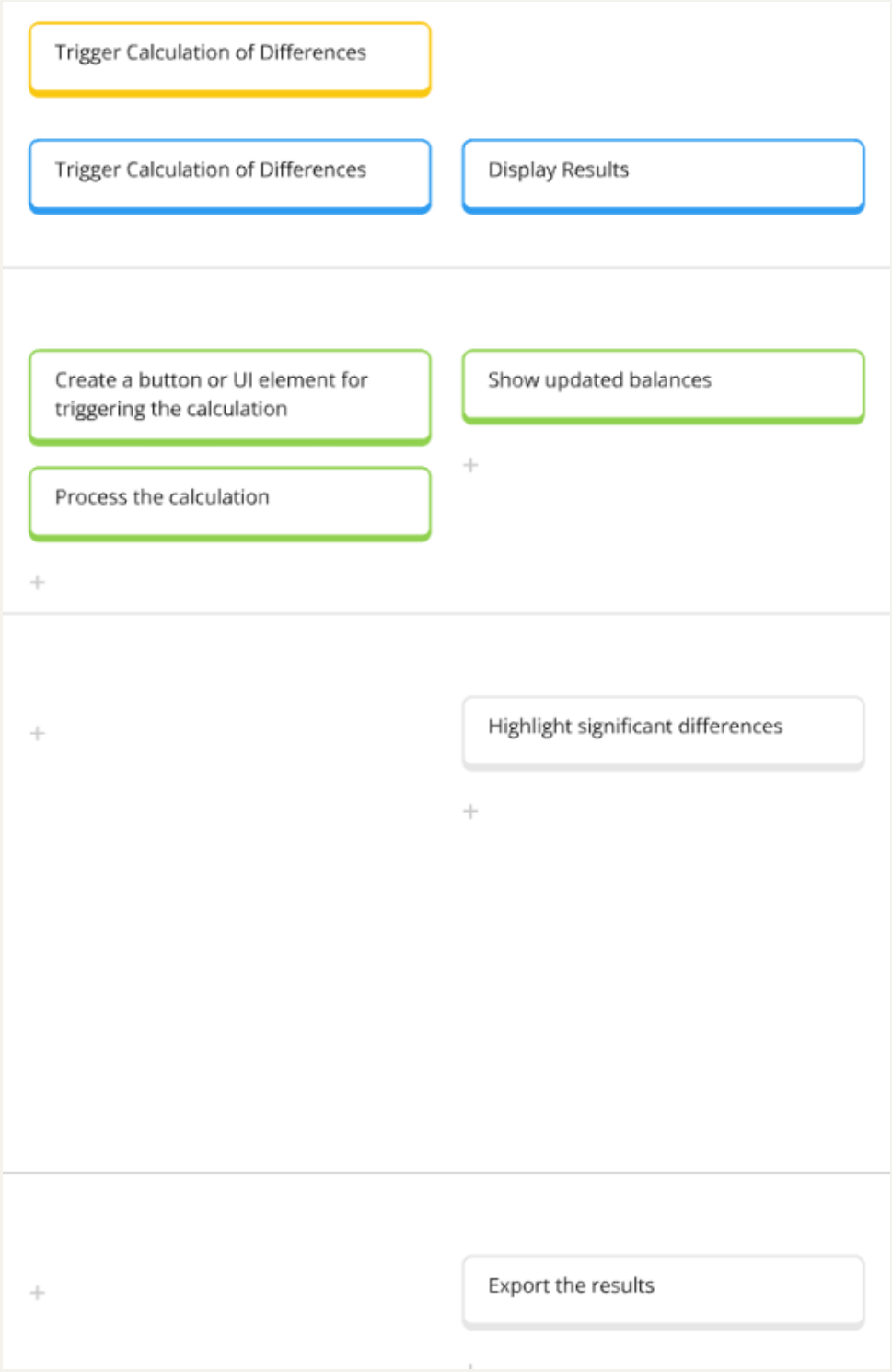

Figure 8: Trigger Calculation of Differences

4 Domain Model Overview

Domain Model Overview

Core Entities:
Subscriptions: Links co-producers to subscription periods. Connected to selected product offers and delivery dates.
SubscriptionPeriods: Defines periods with a name, start date, and end date. Associated with delivery dates and product offers.
ProductOffers: Defines offers for specific subscription periods, including payment methods and modes. Linked to products.
Products: Includes product details such as name, description, price, and delivery units. Connected to product types and producers.
DeliveryDates: Represents specific delivery dates within a subscription period.

Relationships:
Subscriptions <-> SubscriptionPeriods: Subscriptions are tied to a specific period and involve delivery dates.
ProductOffers <-> Products: Product offers are based on the available products from producers.
Products <-> ProducersInfo: Producers provide the products, establishing ownership.
Subscriptions <-> SelectedProductOffers: Tracks the specific products chosen within a subscription.
SubscriptionPayments: Records payments tied to subscriptions and selected product offers.
CoproducersInfo <-> Subscriptions: Co-producers are associated with subscriptions.

Supporting Entities:
AspNetUsers and Related Tables: Manages user authentication and authorization with roles, claims, and tokens.
CheckingAccounts: Maintains balances for users, linked to payments and financial management.
CompoundProductProducts: Tracks relationships between composite and individual products.

Relationships Summary:
The domain model highlights a tightly integrated system where users (producers, co-producers) manage subscriptions and product offers within defined periods. Delivery dates, product details, and payments are all interconnected, ensuring the flow of subscriptions is accurate and traceable. This structure ensures flexibility for product offerings while maintaining financial accountability and user authentication.

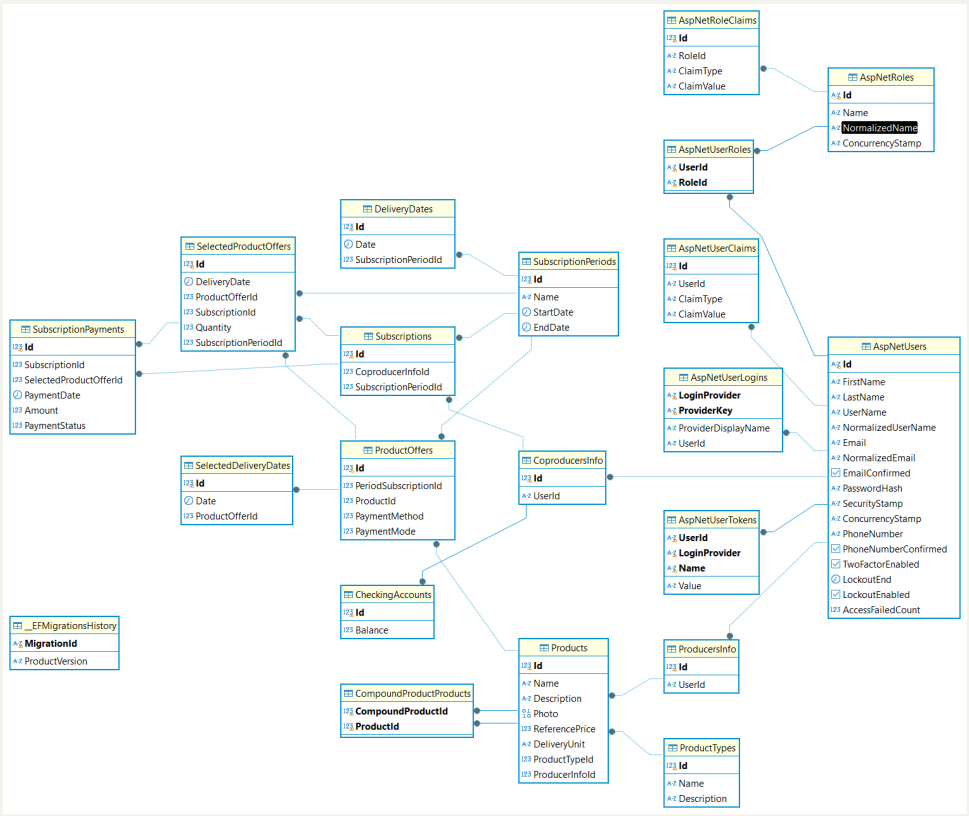

Figure 9: Domain Model Diagram

5 Technologies Used

Technologies Overview

Backend:
The backend is built using .NET 8.0 with `C#` as the primary programming language. The architecture follows the Onion Architecture principles, ensuring a clean separation of concerns and maintainability.

Frontend:
The frontend is implemented using the Model-View-Controller (MVC) pattern. Technologies used include HTML, CSS, and Bootstrap for responsive design and user interface styling.

Database:
PostgreSQL is used as the database management system, providing reliable, scalable, and robust support for handling complex queries and ensuring data integrity.

Key Benefits:
- The use of .NET 8.0 provides robust performance and compatibility with modern development standards.
- `C# Programming` ensures efficient and scalable backend logic.
- PostgreSQL offers advanced data management features and excellent performance for complex queries.
- Onion Architecture promotes testability and flexibility in system design.
- MVC on the frontend ensures a structured approach to user interface development.
- Bootstrap enhances responsiveness and improves the overall user experience.