CHRISTMAS GIFTS ORDERING SYSTEM (XGOS)

1 Contents

2 Assumptions 3
3 System components overview 3
4 Basket processing sequence 4
5 Domain data model 5
6 High level system design 6
7 Technology stack 7
8 High availability and scalation. 7
9 Integration 7
9.1 DigitalWish 7
9.2 DeedsAndDonts 8
9.3 Provider venues 8
9.4 Sledgistics 9

2 Assumptions

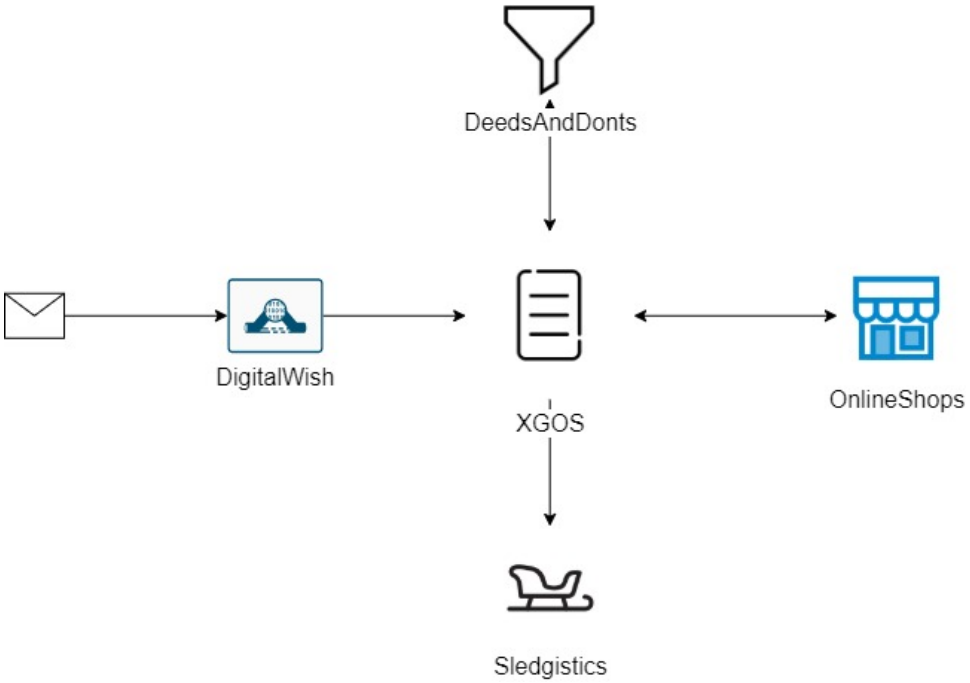
The proposal is based on a few assumptions not mentioned explicitly. Among others:

- DigitalWish would be an order entry system. It can generate a unique basket id across time.
- DeedsAndDonts is a simple compliance module. It has only two possible responses for a kidId: approve or reject.
- Sledgistics input is just a basket with the list of orders and delivery reference.
- There is some sort of central repository, with unique identifiers shared by the system, such as kidId and productid.
- Online venues which fill online orders are treated as exchange venues and extracted from XGOS responsibilities. XGOS magically knows where to best send the orders and will pick the best venue. Its only responsibility is to send the order with central productid and wait for a filled order.
- The simplest scenario is the only one considered. Response to failures such as a potential retry or resend will not be specified.

3 System components overview

XGOS system interacts with the following components defined in the specification:

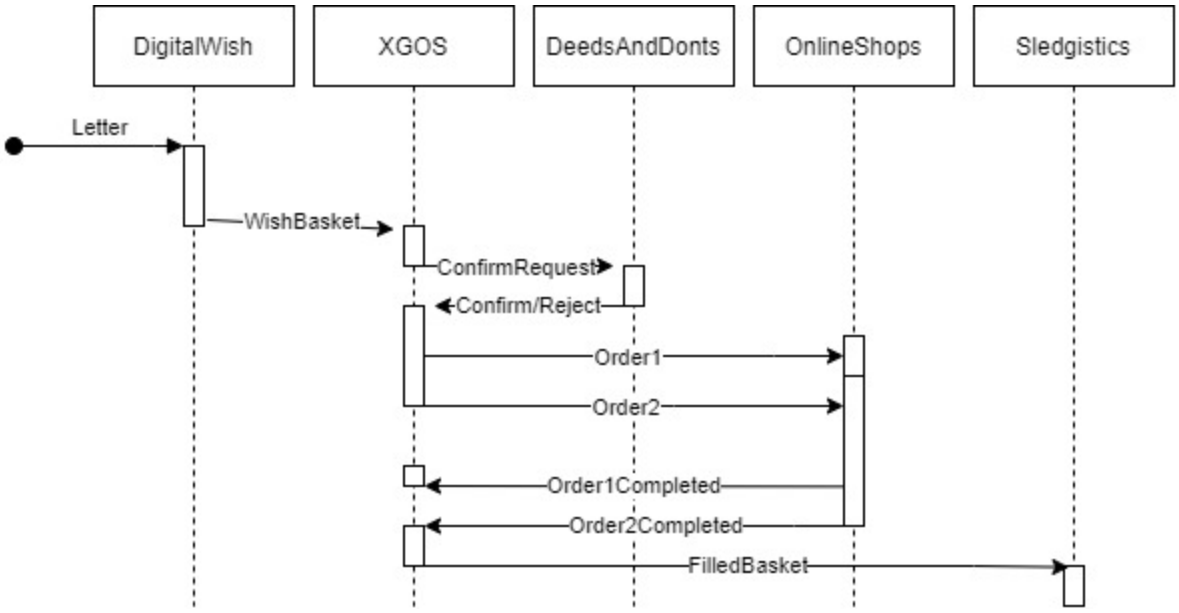
- DigitalWish: Process input letters and other channels, transforming them into an executable basket of presents.
- DeedsAndDonts: Approves or rejects the processing of a basket.
- OnlineShops: Component which interacts with online venues.
- Sledgistics: Logistics component for route planning and sledge-packaging.



XGOS is the central component in the diagram above, processing baskets interacting with all the described components. In the next section, the business flow is described.

4 Basket processing sequence

The following sequence diagram describes the proposed business flow:

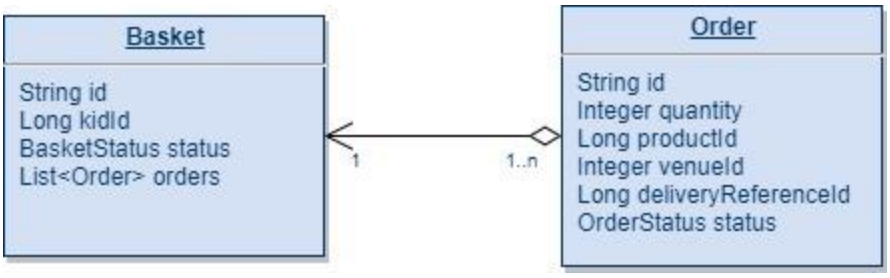


- DigitalWish creates the WishBasket from the input source, containing a kid id and the list of products to purchase and quantities.
- XGOS pulls the messages and sends a ConfirmationRequest with the kid id to DeedsAndDonts, which is asynchronously approved or rejected.
- Upon rejection, XGOS cancels the basket and no further processing is performed. On the other hand, if the message is confirmed XGOS sends the basket orders to the most appropriate online shops.
- Orders are asynchronously completed. Once all basket orders are completed, a FilledBasket is sent to Sledgistics with all the relevant information.

5 Domain data model

The business domain within XGOS is very simple:

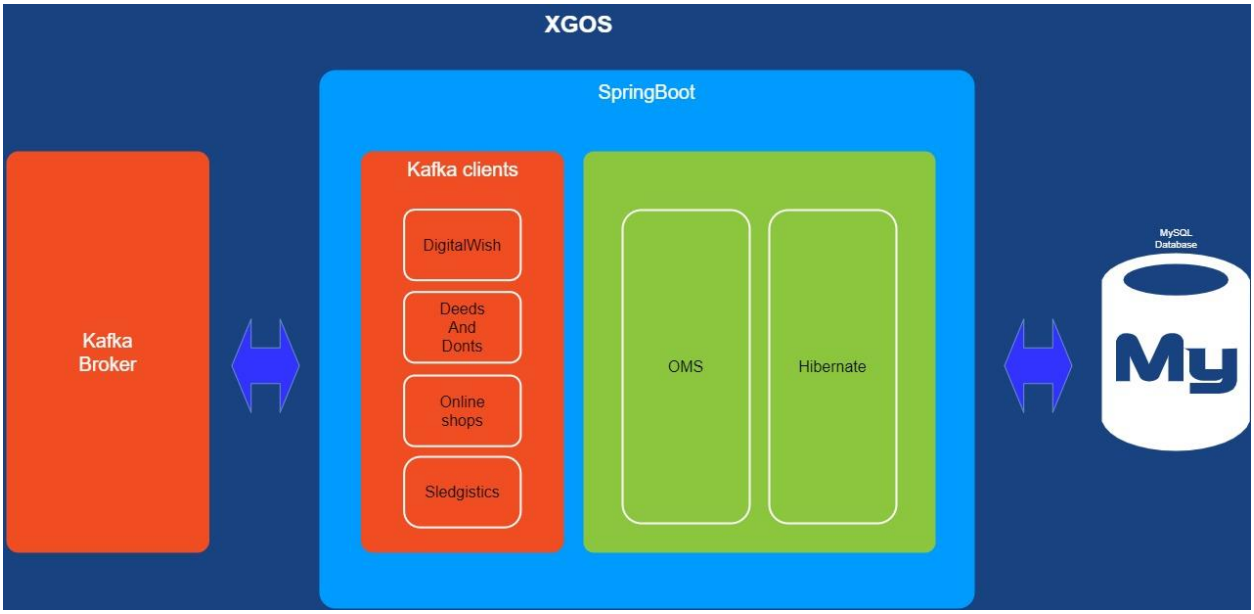
- A Basket object to keep track of approval and confirmed orders.
- The orders sent to Online Vendors with status.



6 High level system design

XGOS will only need to interact with a database and different Kafka topics to consume and produce messages.

Kafka is used as messaging layer to communicate with the rest of the systems. A database will be needed to maintain the state of the Baskets.



7 Technology stack

The proposed technology stack is the one shown on the previous section:

- Kafka can be used as a messaging layer. This will allow horizontal scalability, as data can be distributed easily without data loss.
- We need to keep the state of the basket and orders, hence we can use an ORM like Hibernate with a database like MySql, which like many others provides high availability functionality.
- For the XGOS process itself, we can use SpringBoot, as provides easy integration with all them.

8 High availability and scalation.

Using Kafka as a message layer provides an easy way to scale horizontally. We can partition WishBasket topic and spin off additional XGOS instances adding more consumers. Several partitions for incoming WishBaskets should be created. More consumers can then be distributed across additional instances during peak times or concentrated in a single node during quiet periods. Additional instances of MySql and OnlineShops can be added linked to new XGOS. Each XGOS instance should have its own topic to DeedsAndDonts to ensure no additional messages are read.

On the database side, we can use async replication provided by MySql. On a potential recovery scenario, some data loss might be expected, but should be able to recover WishBaskets from the Kafka streams. This might need to some duplication with DeedsAndDonts, OnlineShops and Sledgistics, so some synchronization might be needed to avoid duplicates if this is not acceptable.

Assuming between 3 to 4 presents per kid, we might need to process 10 billion orders. The current domain is minimal and likely to grow, but we can assume 1k bytes per order -raw data- being conservative with the current design. That sums up to 10Tb of raw data plus a couple of indices, so several MySql servers will be needed. Special care should be taken when interacting with the database to make sure the solution will be able to consume at an acceptable speed.

9 Integration

The integration with the rest of the components is done with Kafka streams as mentioned. The following section describes a draft of the messages sent and received by each component.

9.1 DigitalWish

Producer of WishBasket, with embedded Wish objects and the kidId.

Object	Field	Type	Description
WishBasket	id	String	Unique id for the basket
	kidId	Number	Unique id for the kid
	wishList	[Wish]	List of Wish objects

Object	Field	Type	Description
Wish	quantity	Number	Quantity to purchase
	productId	Number	Unique product id.

9.2 DeedsAndDonts

Consumes of ConfirmationRequest and produces a RequestResponse with the verdict.

Object	Field	Type	Description
ConfirmationRequest	basketId	String	Referenced WishBasket id
	kidId	Number	Kid id

Object	Field	Type	Description
RequestResponse	basketId	String	Referenced WishBasket id
	kidId	Number	Kid id
	approved	Boolean	True/false if the kid id was approved or not

9.3 Provider venues

The different provider venues for Amazon, Ebay and AliExpress consume Order objects and produce a response OrderFill with a reference id to locate the item once the Order is filled.

Object	Field	Type	Description
Order	id	String	Unique order id, can be formed out of the original WishBasket id for better reference
	quantity	Number	Quantity to purchase

	productId	Number	Unique product id

Object	Field	Type	Description
OrderFill	id	String	Original order id
	quantity	Number	Purchased quantity
	productId	Number	Unique product id
	deliveryReferenceId	String	Delivery reference for collection

9.4 Sledgistics

Consumes the final FilledBasket with the list of items and location.

Object	Field	Type	Description
FilledBasket	id	String	Original WishBasket id for better reference
	kidId	Number	Unique kid id
	orders	[Order]	Array of Order objects

Object	Field	Type	Description
Order	id	String	Provider venue order id
	quantity	Number	Delivered quantity
	productId	Number	Unique product id
	deliveryReferenceId	String	Delivery reference for collection