Tasty Trails

Solution Design Proposal

Document Information	
Version	1.0
Authors	Dušan Nikolić
Date	8 Sep 2024

Table of Contents

Executive Summary

Tasty Trails is a restaurant chain available in two countries and at over 140 locations. Tasty Trails' is to provide superb customer service with diverse offers and affordable prices, all while creating beautiful and tasty food.

This document represents the output of the system design analysis phase and its purpose is to provide an overview of assumptions, risks and non-functional requirements that were taken into consideration when designing the system architecture.

The system architecture is available as a separate document, which contains a high-level design of the system, as well as a more detailed design of the major system components.

Assumptions

This section is intended to depict the assumptions with which in mind the system was designed.

Number of Daily Orders

Assuming that the Tasty Trails restaurant chain operates only in mid-large cities (there are usually many competitors in mid-large cities, but there are also usually multiple instances of the same restaurant chain in one city), and its peak hours are in the evening, we could say that **each distinct** restaurant counts roughly 200 orders per day.

This roughly sums up to:

70 restaurants x 2 locations x 200 orders = ***28000 orders per day***

Zooming out a little, this calculation gives us:

28000 daily orders x 30 days in a month = ***840000 orders per month***

Rate Limiting

The API Gateway controls the rate at which API requests are let through. The limit depends on the target service:

- *Order Management API* - up to 3 orders per minute per user
- *Order History API* - up to 30 requests per minute
- *Price and Offer API* - up to 30 requests per minute
- *Customer API* - up to 30 requests per minute
- *Partner Agreement API* - up to 20 requests per minute
- *Inventory API* - up to 30 requests per minute

These limits should be monitored and adjusted accordingly.

Risks

Data Synchronization and Consistency

Due to the distributed nature of the system, there is a risk of data synchronization and consistency if the synchronization service fails or is not able to keep up with the request rate.

This risk could be mitigated by implementing smart monitoring and alerting mechanisms, as well as automatic retry strategies to lower the risk of data inconsistencies.

Network Failures

Prolonged network outages could disrupt the system, causing outdated or stale offer, price and inventory information. Also, online customers placing orders would experience delay or service unavailability due to the central system not being able to connect to a local restaurant node.

This risk could partially be mitigated by designing the system so that local restaurant nodes can operate offline for a short period of time and synchronize with the central system when the network connection is restored.

Dependency on External Services

Tasty Trails SLA could be impacted by dependencies to external systems. If the payment gateway was to go down, in-restaurant customers would not be able to complete their orders. If the supply provider system experiences downtime, the restaurants could be at risk of not having enough supplies to meet the customer demand.

These risks could be mitigated by allowing customers to complete orders by temporarily storing the payment information locally and synchronizing with the payment gateway upon its availability. Also, the risk of supply providers going down could be mitigated by requesting greater supply levels in advance. Both of these risks could also be mitigated by considering alternative providers.

User Experience

Overcomplicated, unintuitive or simply poor UI/UX on SSK devices could be confusing and frustrating to customers, which could cause incomplete orders or orders that take too much time to complete, ultimately leading to unsatisfied customers. The user experience could also be negatively impacted by unpleasant or slow response from the delivery partners, leading to downvotes on partner platforms.

This risk could be mitigated by employing a UI/UX designer to create simple and intuitive SSK UI design. This could also be mitigated by performing A/B testing to analyze user behavior and

make better decisions on UI/UX changes. Negative user sentiment regarding the delivery partners could be mitigated by agreement termination.

Architecture Characteristics

This section highlights the main architecture characteristics that were considered when designing the system architecture.

Scalability

The primary business metric for the Tasty Trails system is the **number of food orders per day**. Tasty Trails operates in two countries and counts a total of 140 restaurants (70 per country). Knowing that Tasty Trails is open for integration to many different delivery services, we may assume that the number of users per day is significant.

High Availability

The system should be highly available due to its hybrid hosting nature. The cloud provider guarantees an SLA of 99.99% (four nines), meaning the system downtime amounts to:

- **Daily: 8.6s**
- **Weekly: 1m 0.48s**
- **Monthly: 4m 21s**
- **Quarterly: 13m 2.4s**
- **Yearly: 52m 9.8s**

Although not covered by the initial design blueprint, the stakeholders have plans to implement monitoring and control services that would allow remote control of SSK devices in case of a failure. Once implemented, this will have a significant impact on infrastructure and operations cost savings.

Fault Tolerance and Business Continuity

Due to the distributed nature of the system, a decision was made to host local restaurant nodes on-premises. With this approach, the system should be resilient enough in case of network disruption with the central system. Each restaurant node is capable of storing order requests locally and can synchronize with the central system when it is back online. The local restaurant node directly integrates with the payment provider, allowing customers to complete payment even when the central system is offline.

Security

The system uses secure transfer protocols and all communication originating from the outside world is routed through the API Gateway. Payment transactions are handled by an external payment gateway, therefore sensitive information such as credit card numbers are not stored.

Architecture Decision Records

This section is intended to justify selected technologies and/or approaches used in the Tasty Trails system design.

Architecture

The system will rely on a microservices architecture with a message bus in the heart of the system enabling asynchronous communication and synchronization between major components.

Deployment Model

The system will be hosted using a hybrid deployment. This is partly because the system is highly distributed, with over 140 locations requiring local hardware and software. Therefore, software components used in the restaurant will be hosted on premises, whereas the central management system will be hosted in the cloud, specifically Azure.

Frameworks and Programming Languages

Large part of the system consists of REST APIs which will be built in ASP.NET and C#. The reason behind this decision is the fact that the entire system stack is based on Microsoft technologies, the development team is highly skilled in this tech stack and the stack is simply very mature and stable, therefore this was a natural choice.

API Gateway

This will be the main entry point into the system for any client application, including partner channels. The API gateway will be used to authenticate users entering the system, as well as for URL based routing to appropriate backend services.

Message Bus

Because the local restaurants need to constantly be in sync with the central system, Azure Service Bus was selected as a Message Bus to handle asynchronous communication between the services.

Scheduled Jobs

Quartz.NET will be used to handle scheduled jobs within the local restaurant nodes. On the cloud side, Azure Functions with *TimeTrigger* will be used to execute periodic tasks.

Relational Databases

Usage of relational databases, namely PostgreSQL for the local node part of the system and Azure SQL for the cloud part, is sufficient and desired for the OLTP processes. In case the stakeholders wish to support OLAP processes, an alternative to relational databases should be considered.

Blob Storage

This is required for storing raw files such as partner agreement documentation. Although the system design does not specifically mention reporting and invoicing services, this type of storage could well be used to store those files as well.

Data Synchronization

Due to the nature of asynchronous communication between the local restaurant nodes and the central system (via the message bus synchronization service), eventual consistency should be sufficient for most operations, such as offer and price updates. However, strong consistency should be enforced for order and payment operations to avoid discrepancies and failed transactions.