## Possiveis Perguntas

## Inbound/Kafka:

* Why is it possible to create Advanced Shipment Notices (ASNs) without any Purchase Order (PO) associated?
  + Every time new cargo arrives in a warehouse it must always have an ASN associated and it's possible for a shipment to be unrelated to any purchase (just for stock relocation purposes for example) so it makes sense to be able to create independent ASNs.
* Why does an independent ASN 's page follow the format of the PO ‘s detail page, even though it doesn’t have any PO information to display?
  + For functional and visual coherence purposes. This way, from the user’s perspective the experience is more pleasant and intuitive.
* Why is the creation of ASNs done by uploading an Excel file instead of a form?
  + The digitalization of ASNs in Excel files is an industry standard, for its simplicity and portability aspects.
* How did you integrate with Kafka and what benefits does it provide?
  + We integrated with Kafka in our Inbound microservice to enable real-time streaming of data between our microservices. Kafka is a distributed streaming platform that is designed for high-throughput and low-latency data processing, and it is well-suited for handling large volumes of data in real-time.
  + To integrate with Kafka, we used the Kafka client library for the programming language that we used to build our microservice. This library provides APIs for producing and consuming messages from Kafka topics, and it allows us to easily connect our microservice to Kafka and start streaming data.
  + Some of the benefits that Kafka provides for our e-commerce app include:
    - Real-time processing: Kafka allows us to process data as it is generated, rather than having to batch process it later. This makes our app more responsive and efficient.
    - Scalability: Kafka is designed to handle high volumes of data and can be easily scaled up or down as needed. This allows us to handle large amounts of traffic without sacrificing performance.
    - Reliability: Kafka provides built-in fault tolerance and data durability, which helps to ensure that our app stays up and running even in the event of a failure.
    - Flexibility: Kafka allows us to easily connect different microservices and systems, which makes it easier to build a flexible and modular architecture."

## Catalog:

## Brand Config:

## Devops:

* Can you describe your deployment and continuous integration process?
  + XXX

## Perguntas Gerais:

* How does your app address the needs of Maersk's business?
  + Our app addresses the needs of Maersk's business by providing a reliable and easy-to-use platform for managing stock and other products. It offers features such as real-time inventory management, customizable product catalogs, and flexible pricing options, which can help Maersk to streamline its operations and improve its bottom line.
* How does your app integrate with Maersk's existing systems and processes?
  + Our app is built using the same technologies used by Maersk. It follows the same architecture in microservices with communication in Kafka. And it’s ready to receive and send Kafka messages to communicate with other microservices.
* How have you ensured the security and reliability of your app?
  + We did not focus on security and authentication in the development of our app. However, we understand that these are important considerations for any e-commerce platform, and we are committed to implementing robust security measures to protect our users' data. To address this issue, we recommend implementing SSL encryption to protect data transmitted between the app and the server, as well as using password hashing and two-factor authentication to secure user accounts. We also recommend implementing role-based access controls to ensure that users only have access to the features and data that they are authorized to see.
* How do you envision Maersk using your app on a day-to-day basis?
  + Maersk might incorporate our code in pre-existing one, or use any of our microservices with some adjustments to fit their overall system.
* How have you tested your app and what kind of testing have you done?
  + We tested our app using a combination of unit tests and integration tests. We also did manual testing to ensure that the app was easy to use and had a good user experience.
* How will you provide ongoing support and maintenance for your app?
* What programming languages and frameworks did you use to build your app?
  + For the backend we used C# and the “.NET” framework. For the frontend we used Flutter. These were chosen because Maersk is in the processing of migrating to said technologies.
* How did you design and architect your app, and why did you make certain design choices?
  + We followed Maersk’s architecture dividing the app in microservices each with its own backend and frontend. We also ensured that the data was coherent across the multiple backends and used a common design system for the fronteds. The design system consisted of having a bank of widgets following a theme (with defined colors and fonts) from which all microservices would retrieve the assets. This leads to a consistent and pleasant look across all the microservices. By separating the app in microservices we avoid having a single point and achieve modularity.
* How did you implement your microservices and why did you choose a microservices architecture?
  + We followed Maersk’s architecture dividing the app in microservices each with its own backend and frontend. By separating the app in microservices we avoid having a single point and achieve modularity. So if a microservice fails, the others can keep on working independently.
* Can you describe your app's database schema and how it is optimized for performance?
  + XXX
* How did you handle scaling and load balancing for your app?
  + As of this moment, the platform relies upon the distribution of services (and therefore workloads) throughout different microservices. This creates fault tolerant capabilities, and allows the system to scale when new services are requested from the customer, and later implemented with ease. The .NET infrastructure provides reliability when handling a large number of requests. Future iterations of the project can place a standard load-balancer with ratelimiter capabilities (nginx e.g.) to control the request flow to base service.
* What programming languages and frameworks did you use for the backend of your project, and why did you choose them?
  + For the backend of our project, we used C# and the .NET framework. We chose these technologies because they are powerful and widely used, and they offer a range of tools and libraries that make it easier to develop and deploy applications. C# is a particularly popular choice for building web applications, and the .NET framework provides a solid foundation for building scalable and reliable apps.
* What technology did you use for the frontend of your project, and why did you choose it?
  + For the frontend of our project, we used Flutter. We chose Flutter because it is an open-source framework that allows us to build natively compiled apps for mobile, web, and desktop from a single codebase. It uses the Dart programming language, which is easy to learn and use, and it provides a range of tools and libraries for building user interfaces and interacting with the backend.
* What database did you use for your project, and why did you choose it?
  + We used Postgres for our database. We chose Postgres because it is an open-source relational database management system that is widely used for building web-based applications. It is known for its reliability, performance, and support for a wide range of data types and SQL features, which made it a good fit for our project.