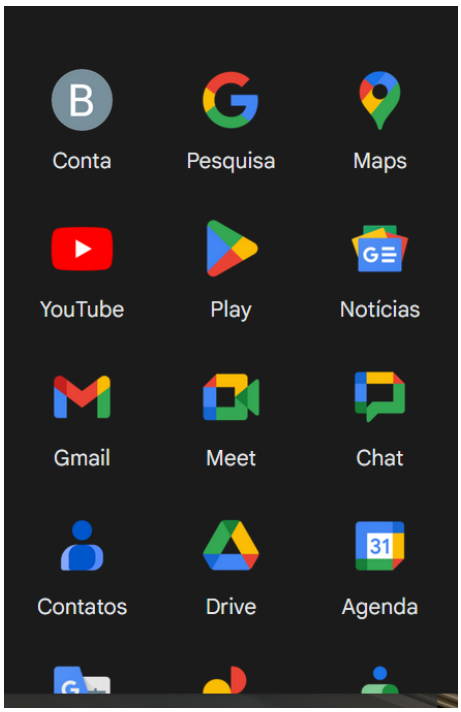
Challenge 2 - Development

1. **How would you structure the architecture and stack of this platform?**

Architecture

The planned structure for this platform use a cloud approach with microservices. As there are several technological services and products to deliver to the end user in a single place, we can offer a portal structure similar to what Google delivers in the image below:



This way, we would aggregate the most diverse services into a single portal, where each service/product would have its own structure of frontend, backend, cloud infrastructure, CI/CD, cache, monitoring, logs, data and scalability. An intermediary service would be responsible for generating insights from one platform to another in order to improve other products through information analysis and intelligence. Therefore, we can distribute it as follows:

1. **Frontend:**
    - Modern frameworks that optimize development and allow an intuitive and user-responsive approach. Options: Angular, React, or Vue.js. I would use Angular
2. **Backend:**
    - Microservices architecture with well-defined responsibilities (authentication, specific integrations, CRUD by product/functionality segment, financial transactions) with Python, Node or Java and transactional processing and communication between services through queues such as Kafka or RabbitMQ (both will allow us process streaming data for some platform features, such as financial transaction volume)

    - Use of OpenTelemetry thinking about the observability of each service, it is complex to test, debug and find unexpected behaviors in microservices architectures
3. **Database:**
    - Relational database (Oracle, PostgreSQL) for structured information such as user registrations, credit information, balance, accounts, billing.
    - NoSQL database for user preferences, profile, booking history, descriptions, photos, reviews (data that can be more changeable in terms of structuring.
4. **Authentication and Authorization:**
    - Unified authentication between all platform services with OAuth and JWT, for example.
5. **Cache:**
    - Redis or Memcached to improve the speed of frequent searches
6. **Internal and external integrations:**
    - APIs Rest para extração, inserção e manipulação de dados entre cada um dos serviços/produtos (sistemas de gerenciamento de propriedades, serviços de fornecimento de dados, serviços de gerenciamento de receita, etc)
7. **Infrastructure, security and monitoring**
    - Use of cloud services with redundancy flexibility in mind. AWS for Core processing of applications, due to the set of services available: Glue, EMR, Kinesis, S3, Lambda, Athena. Also noteworthy are the monitoring services and managed services on AWS, which bring quality and peace of mind to IT teams.
    - Simplified and independent deployment per service and even modules of a service with the implementation of Docker and Kubernetes.
    - Implementation under HTTPS and input validation.
    - Open telemetry for observability of all services
    - Zabbix/Prometheus with presentation in Grafana (generation of user-friendly insights/analysis)
    - Automation and independence in the wake of deployments with CI/CD per module and functionalities within these modules, for example API does not need to follow the same versioning of product registration, nor billing, therefore we would have different CI/CD tracks and specific configurations in the Gitlab.
    - API Gateway for managing traffic and communications, in addition to providing rules and restrictions as a layer of protection.

Although we have already introduced some non-functional requirements above, it is worth highlighting the concern and definition under the following aspects: performance, reliability, availability, documentation, load testing, usability, efficiency, regulatory compliance and costs for the architecture model versus use (there is an interesting case about Prime Video, where after evaluating consumption/limits and use in practice (and not in theory)(example of observability), they switched to a monolithic model (also thinking about costs)

2. **What other product would you use as a benchmark for our product?**

Considering that the platform has several services to be incorporated into the portal, we can highlight benchmarks for the other companies below, each with different aspects:

1. **Expedia:**
    - Section dedicated to vacation rentals. It can offer insights into the integration of different travel services and the user experience.
2. **Kigo:**
    - Property management system for vacation rentals. It can offer insights into task automation and operational management.
3. **HomeToGo:**
    - Vacation rental search engine. It can provide insights into how to aggregate and present information from different sources.
4. **TurnKey Vacation Rentals:**
    - Company that manages seasonal rental properties. Can provide insights into property management services and quality standards.
5. **Beyond Pricing:**
    - Price optimization platform for vacation rentals. It can offer insights into advanced pricing strategies.
6. **Wheelhouse:**
    - Dynamic pricing platform for rental properties. Can offer insights into ongoing price optimization
7. **TripAdvisor Rentals:**
    - It has a section dedicated to vacation rentals. It can provide insights into the importance of user reviews and feedback.
8. **Vrbo (Vacation Rentals by Owner):**
    - Vacation rental marketplace focusing on entire properties.

3. **What will be the main technical challenges in developing this product?**

The main technical challenges for the product include: Integration with different internal and external systems and services, scalability, data privacy and security, user experience on different devices (there is now a wide range of products in the portfolio, payment processing reliably, compliance with laws (requirements), continuous integration, team capable of evolution and maintenance, complexity of intelligence algorithms, continuous integration and effective implementation.

The challenges involve a multidisciplinary approach, which involves software engineering activities, security, design, user experience, feedback, testing, agile methodology, statistics and mathematics with refinement of models and parameters for data intelligence, in addition to the science involved in data.

4. How would you structure the development team(s) for this product?

The number of people on the team and team seniority may vary in relation to cost and time, however for the team to be complete, we need front-end developers, back-end developers, UI/UX designer, QA/Testers, Software Scientists, Data, System Administrators, Information Security Administrators, Support/Operations Team

Specification of the Development team structure to develop all types of products listed question 2 that will make up the portal:

- 16 Back End Developers
- 5 Front End Developers
- 3 Data Scientists
- 4 QA/Testers
- 3 Scrum Master
- 3 PO (or Committee of POs)