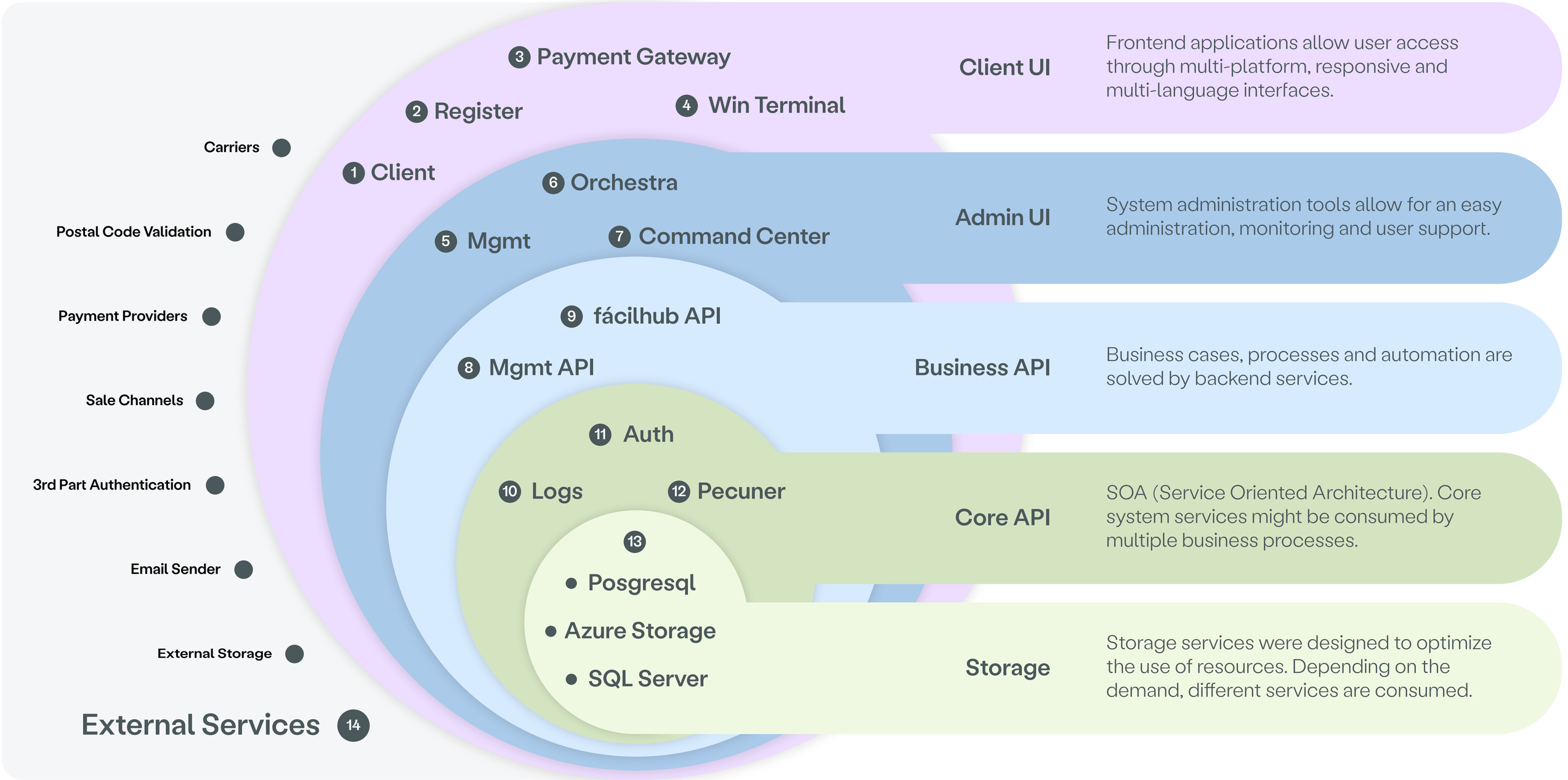
fácilhub

System Architecture



In overall terms, **fácilhub** enables receiving orders from multiple sales channels, assists in fulfillment and then obtains a carrier label.

To achieve this, the system has multiple services, at different levels, designed to optimize the resources consumption, ensure data integrity, and scale up and out; leveraged on Microsoft Azure services.

The following is a brief description of the system's architecture.

Users can sign themselves up to **fácilhub** via the Register application **(2)**. When a new company is added, the app automatically creates an account and assigns the requested user the role of administrator. Then, he/she will be able to create new profiles and assign them to the following users.

fácilhub clients operate on the Client application **(1)**. This is a 100% responsive and multi-language progressive web application (PWA) that has been developed with special attention to UX to have a consistent interaction and ensure fast user adoption.

Our application enables, among other functionalities:

- Visualize key information on a dashboard.
- Obtain orders from multiple sales channels.
- Fulfill orders
- Compare rates
- Print labels
- Track shipments
- Audit accounting
- Inventory management
- Generate reports
- Manage:
    - Sales channels
    - Payment methods
    - Warehouses
    - Users and Profiles

Win Terminal **(4)** is a native Windows application that enables unrestricted access to local devices. This facilitates silent and batch label printing, volumetric and weight data collection from packages.

To register a payment method, Payment Gateway **(3)** is an application that protects the user's data, preventing it from being stored on any other system outside the payment provider. Once the payment method has been validated, the account is ready to operate.

All the business logic mentioned so far is supported by **fácilhub** API **(9)**, which also communicates with External Services **(14)** such as: Postal Code Validation; e-mail Sender; External Storage; among others. This service is tasked with linking the data transfer between multiple sales channels and carriers.

Management **(5)** is a web application reachable via intranet that allows managing customer accounts and their settings, including:

- Account registration and cancellation
- Carrier assignation
- Rate set management
- Sales prices and surcharges

Orchestra **(6)** is another administration tool that enables the management of low-level services.

- Monitoring processes
- Access system log
- Manage account credit in Pecuner **(12)**

Command Center **(7)** is an application health status monitor. From there, it is possible to visualize incidents, failures, and the workload of each service, enabling early actions to be taken to minimize infrastructure and other risks.

The backend of the 3 previous applications is supported by Management API **(8)**. This is responsible for integrating the base services with the business layer. It also takes part in the communication with low level external services (3rd Part Authentication; Carriers; Sale Channels).

Every data transaction must meet two fundamental conditions that are supported by their respective services:

- Logging. Every transaction must be registered. Logs **(10)** is responsible for storing: calling application, related traces, critical exceptions, and errors.
- Authentication and Authorization. All applications, both frontend and backend, are designed based on Service Oriented Architecture (SOA) and work under the authentication context determined by Auth **(11)**.

Additionally, Auth supports the creation of unlimited profiles with granular assignment of permissions to system functionality. Therefore, the account administrator can define, in detail, which permissions are assigned to each user. It also supports multi-factor authentication and authentication by external methods.

Pecuner **(12)** is the service responsible for accounting credits and debits to the customer's account. It communicates mainly with the payment platform **(3)**. It records transactions for traceability and provides data for reporting and cost auditing.

The lowest level of the system consists of storage services **(13)**. Depending on the required objective (structuring data, storing large volumes, fast access, securing critical information, etc.), different methods or technologies are used for data persistence. This optimizes the use of resources, without affecting performance. Elements in this layer can only be reached by low-level services, generating protection barriers outside the system.