Gacha Collection Beckend

290AA – Advanced Software Engineering

Group:
Ardizzoni Francesco
Del Castello Diego
Prestifilippo Colombrino Mattia
Tortorelli Felice

User

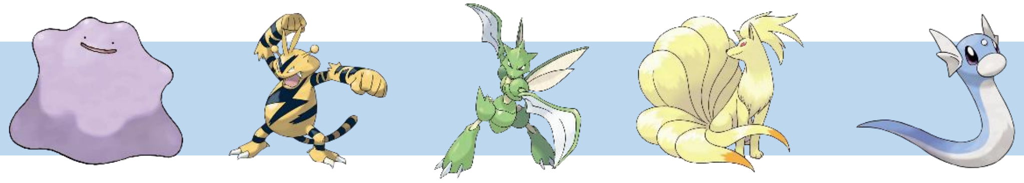
- ID
- First Name
- Last Name
- E-mail
- Password (hash)
- History
- Currency Amount

Catalogue – All 151 Pokémon from 1° gen

- Common (54,45%)

- Uncommon (40%)

- Rare (5%)

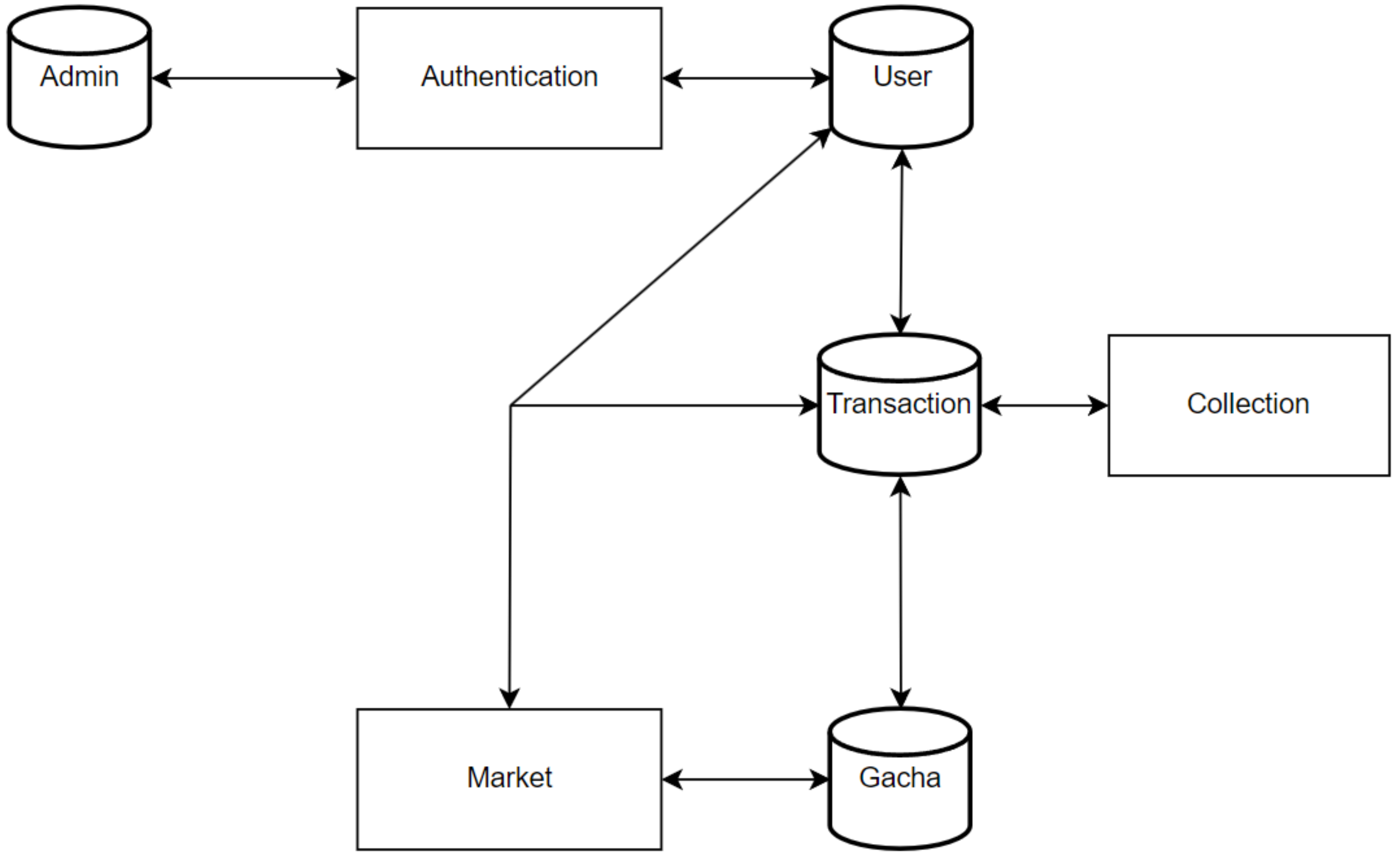


- Epic (0,5%)

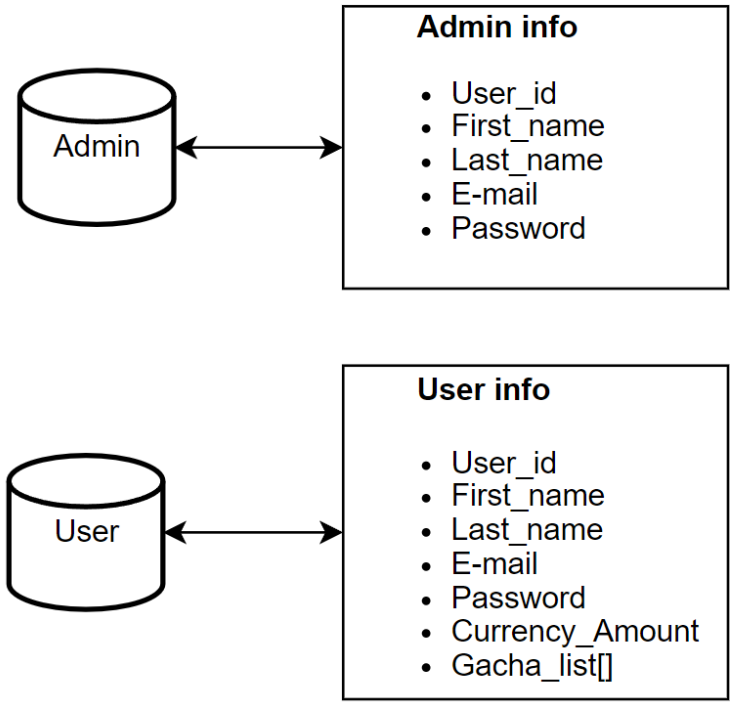
- Legendary (0,05%)

- **In-game currency: Pokedollars** ₽

The Architecture



The Architecture

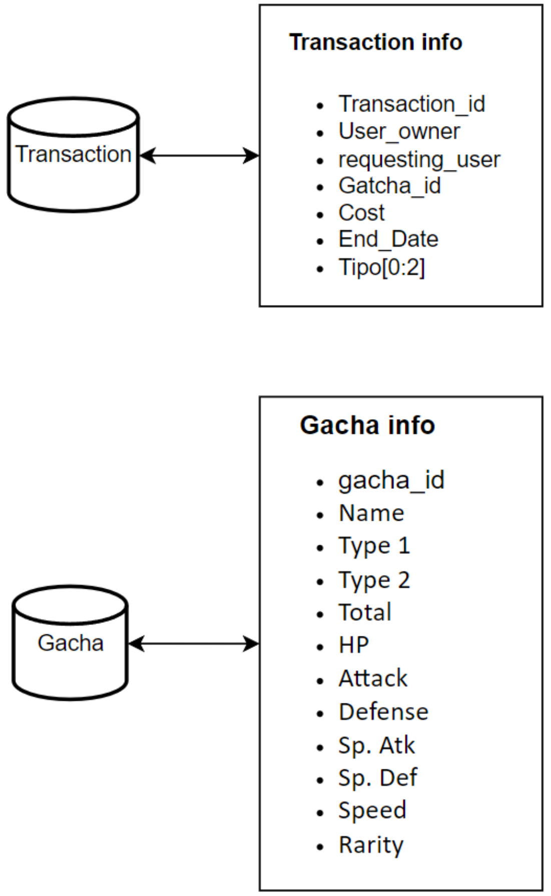


Admin info

- User_id
- First_name
- Last_name
- E-mail
- Password

User info

- User_id
- First_name
- Last_name
- E-mail
- Password
- Currency_Amount
- Gacha_list[]



Transaction info

- Transaction_id
- User_owner
- requesting_user
- Gatcha_id
- Cost
- End_Date
- Tipo[0:2]

Gacha info

- gacha_id
- Name
- Type 1
- Type 2
- Total
- HP
- Attack
- Defense
- Sp. Atk
- Sp. Def
- Speed
- Rarity

The Architecture

Authentication

The user access the authentication to:

- login
- logout
- create a game profile
- delete a game profile
- edit it's game profile

Collection

The user access the collection to:

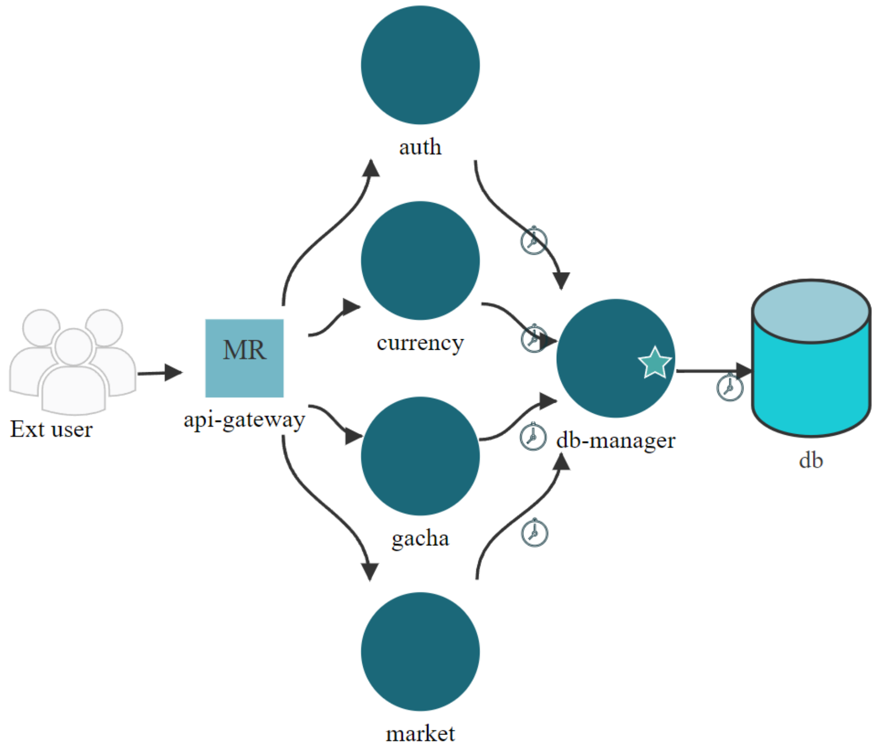
- see their gacha collection
- see the info of a gacha of thier collection
- see the system gacha collection
- see the info of a system gacha

Market

The user access the Market to:

- see the auction market
- set an auction for one of my gacha
- bid for a gacha from the market
- receive in-game currency when someone win my auction
- receive a gacha when I win an auction
- receive my in-game currency back when I lost an auction
- use in-game currency to roll a gacha
- buy in-game currency

Microfreshener



The micro-services

-api-gateway

This API Gateway acts as the main entry point for client requests, forwarding them to the appropriate microservices for processing. It integrates with multiple services.

-db-manager

This API is part of a microservices architecture and handles all the queries to the database.

-auth-service

This API is part of a microservices architecture and serves as the user management service for the application. It provides operations such as register a new user or authenticate a user

-currency-service

This API is part of a microservices architecture and serves as the currency management service for the application. It provides operations for managing in-game currency and rolling gacha items.

-gacha-service

This API is part of a microservices architecture and serves as the gacha management service for the application. It provides operations for managing gachas and retrieving gacha infos.

-market-service

This API is part of a microservices architecture and handles transactions related to gacha itemsand user purchases. It interacts with other services to track and update user transactions.

The register service datapath

The registration-related services are exposed externally to clients on port 8000 via an API gateway. When an HTTPS request is received for user registration, the API gateway forwards the HTTPS request to the auth-service/register on port 8001.

The auth-service itself exposes APIs on port 8001 to handle user registration, login, updating user details, and deleting users. To interface with the database, the auth-service sends an HTTPS request to the db_manager, which provides APIs for database-related operations. These database-related APIs are exposed by the db_manager on port 8005.

Once the registration is successful, without any problems on DB, the db-manager service communicates the registration in the database. The auth-service service will then forward the response obtained to the api-gateway

The login service datapath

The login_user-related services are exposed externally to clients on port 8000 via an API gateway. When an HTTPS request is received for user registration, the API gateway forwards the HTTPS request to the auth-service/login_user on port 8001.

The auth-service itself exposes APIs on port 8001 to handle user login. To interface with the database, auth-service sends an HTTPS request to the db_manager(containing the user's email), which provides APIs for database-related operations. These database-related APIs are exposed by the db_manager on port 8005.

The db-manager performs a lookup through the email provided and returns the corresponding encrypted password if successful. The auth.service service will then perform a check on the password now encrypted by the user and the one received from the db-manager. If the match is positive, the auth-service service will then forward the login to the api-gateway.

The gacha service datapath

The currency service is exposed to external clients through an API Gateway on port 8000.

When an HTTPS request for roll_gacha or buy_currency is received, the API Gateway forwards the request to the currency_service via HTTPS on port 8004. The currency_service is responsible for implementing the business logic associated with these requests.

To interface with the database, the currency_service sends an HTTPS request to the db_manager, which listens on port 8005. The db_manager exposes APIs that handle database queries, enabling the currency_service to retrieve or update data as required.

The currency service datapath

The gacha-related services are exposed externally to clients on port 8000 via an API gateway. When an HTTPS request is received for a service related to CRUD operations for gacha, the API gateway forwards the HTTPS request to the gachaservice on port 8002.

The gachaservice itself exposes APIs on port 8002 to add, modify, delete, and view a single gacha or the entire collection of gachas.

To interface with the database, the gachaservice sends an HTTPS request to the db_manager, which provides APIs for database interfacing services. These database-related APIs are exposed by the db_manager on port 8005.

The gacha service datapath

The currency service is exposed to external clients through an API Gateway on port 8000.

When an HTTPS request for roll_gacha or buy_currency is received, the API Gateway forwards the request to the currency_service via HTTPS on port 8004. The currency_service is responsible for implementing the business logic associated with these requests.

To interface with the database, the currency_service sends an HTTP request to the db_manager, which listens on port 8005. The db_manager exposes APIs that handle database queries, enabling the currency_service to retrieve or update data as required.