GachaAndGames System Documentation

Riccardo Fantasia, Leonardo Pantani, Christian Sabella

November 22, 2024

System Architecture Documentation

System Architecture

Architecture Overview

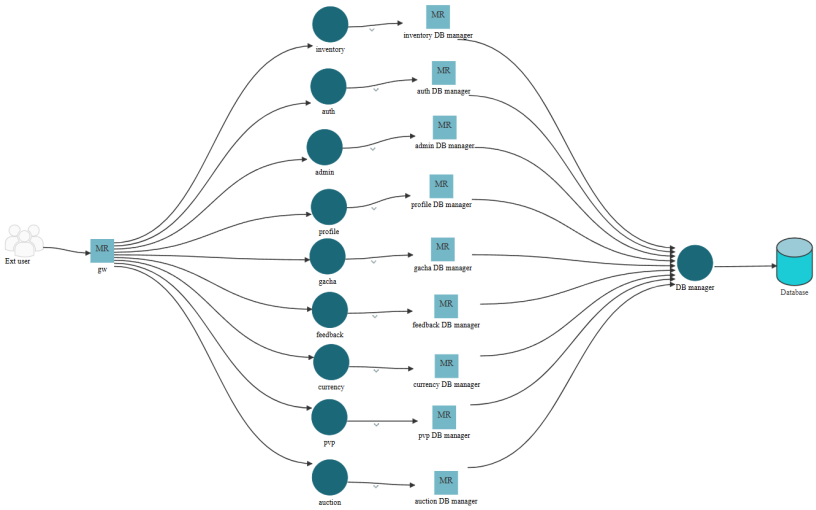

Figure 1: Microservice architecture diagram

Core Services

Auth Service

Service path: `services/auth`

- Manages user authentication through cookie-based sessions.
- Implements secure password hashing and verification.
- Communicates with db_manager for user data persistence.
- Uses circuit breakers for fault tolerance.

Profile Service

Service path: `services/profile`

- Manages user profiles with UUID-based identification.

- Handles profile updates (username, email) and deletions.
- Stores profile information like join date.
- Validates usernames to contain only letters, numbers, and underscores.
- Requires usernames to be at least 5 characters long.

Admin Service

Service path: `services/admin`

- Administrative dashboard for system management.
- Controls gacha pools and item configuration.
- Manages user permissions and roles.
- Views system logs and feedback.
- Can update/delete auctions and profiles.

Auctions Service

Service path: `services/auctions`

- Manages item auctions between users.
- Tracks auction status (active/closed).
- Handles bidding with minimum price validation.
- Records bid history and winners.
- Uses the dbmanager for auction persistence.

Currency Service

Service path: `services/currency`

- Handles in-game currency transactions.
- Manages currency bundles for purchases.
- Supports multiple currency types via codes (e.g., "EUR").
- Validates transaction amounts.
- Tracks user balances.

Inventory Service

Service path: `services/inventory`

- Manages user item collections.
- Tracks item ownership and transfers.
- Records item stats and attributes.
- Stores item acquisition dates.
- Maintains ownership history count.

Feedback Service

Service path: `services/feedback`

- Collects user feedback submissions.
- Validates feedback content.
- Routes feedback to administrators.
- Simple JSON payload with feedback string.
- Session-based user identification.

Gacha Service

Service path: `services/gacha`

- Implements gacha game mechanics.
- Manages item pools with configurable probabilities.
- Four rarity levels: common (50%), rare (30%), epic (15%), legendary (5%).
- Items have attributes rated A-E.
- Validates pool configurations.

PvP Service

Service path: `services/pvp`

- Manages player versus player battles.
- Handles matchmaking.
- Tracks battle results.
- Implemented as Flask service with the dbmanager.
- Uses session authentication.

DB Manager Service

Service path: `db_manager`

- Central database management service.
- Handles data persistence across services.
- Manages MySQL connections and queries.
- Implements retry logic for DB connections.
- Uses environment variables for configuration.

Infrastructure Components

API Gateway

Configuration path: `api_gateway/api_gateway.conf`

- Nginx-based reverse proxy.
- Implements least connections load balancing.
- Configures fail timeout and max fails.
- Routes requests to appropriate services.

Database

- MySQL Database for centralized storage.
- Used by all services through db_manager.
- Configured via environment variables.
- Stores user data, items, auctions, etc.

Player's Example Workflows

Create Auction Flow

Client sends a POST request to /auctions/create to API Gateway:8080. API Gateway routes request to Auctions Service:8080. Auctions Service verifies the session locally with the cookie. The auction price is validated to be between 1 and 1,000,000. Sets auction duration to 10 minutes. Auctions Service requests item details from DB Manager:8080. DB Manager queries MySQL DB:3306 for item details and returns them. Circuit breaker pattern is applied for fault tolerance. If successful, Auctions Service creates the auction. DB Manager inserts the auction details into MySQL DB:3306.

Gacha Pull Operation

Client sends a POST request to `/gacha/pull/{pool_id}` to API Gateway:8080. API Gateway routes request to Gacha Service:8080. Gacha Service verifies if the user has enough credits (minimum 10 credits). Pool details are requested from DB Manager:8080. DB Manager queries MySQL DB:3306 for pool data and returns it. Gacha Service calculates probabilities: common (50%), rare (30%), epic (15%), legendary (5%). Draws item based on the calculated probabilities. User credits are deducted. Gacha Service records the transaction as `gacha_pull`. DB Manager inserts the item details into MySQL DB:3306.

PvP Battle Flow

Client sends a POST request to /pvp/accept to API Gateway:8080. API Gateway routes request to PvP Service:8080. PvP Service verifies session locally using the session cookie. Verifies that the user has at least 7 gacha items per team. Validates item ownership by querying DB Manager:8080. DB Manager queries MySQL DB:3306 to validate ownership. Random stats (power, speed, durability, precision, range) are selected. Potential is used as a tiebreaker if necessary. Logs the match details in JSON format, including gachas_types_used. Processes the battle and determines the winner. Updates winner's pvp_score in DB Manager:8080. DB Manager updates MySQL DB:3306 with the updated pvp_score.

Auction Bidding

Client sends a POST request to `/auctions/{auction_id}/bid` to API Gateway:8080.
API Gateway routes request to Auctions Service:8080.
Auctions Service verifies the auction is still active.
Validates that the bid is higher than the current highest bid.
Checks if the user is bidding on their own auction (not allowed).
Ensures that the user has sufficient funds for the bid.
Refunds the previous highest bidder.
Applies circuit breaker pattern for fault tolerance.
Records the new bid in DB Manager:8080.
Transaction is saved in MySQL DB:3306 as `bought_market`/`sold_market`.

Currency Bundle Purchase

Client sends a POST request to `/currency/bundles/{bundle_id}/buy` to API Gateway:8080.
API Gateway routes request to Currency Service:8080.
Currency Service verifies session locally using the cookie.
Retrieves bundle details from DB Manager:8080.
DB Manager queries MySQL DB:3306 for bundle data.
Processes the payment, handling multiple currency types (EUR, USD, PLN).
Records the transaction in both `bundles_transactions` and `ingame_transactions` tables.
DB Manager inserts transaction details into MySQL DB:3306.

Inventory List

Client sends a GET request to /inventory to API Gateway:8080. API Gateway routes request to Inventory Service:8080. Inventory Service verifies session locally using the session cookie. Inventory data is requested from DB Manager:8080. DB Manager queries MySQL DB:3306 for inventory items. Implements pagination with 10 items per page. Returns the paginated inventory list to the Client via API Gateway.