FEATURE MANAGEMENT SOLUTION PROPOSAL

PREPARED FOR

WorkMotion

PREPARED BY

Rafif Zayed

1. Purpose

A solution that allows us to change the runtime behaviour of applications quickly.

It should support the following features:

- Progressively rollout new features
- Experiment with alternative flows
- Limit features to a set of users or a set of countries
- Shutdown a feature if there is a suspicion of fraudulent activity
- Put a service in read-only mode during maintenance

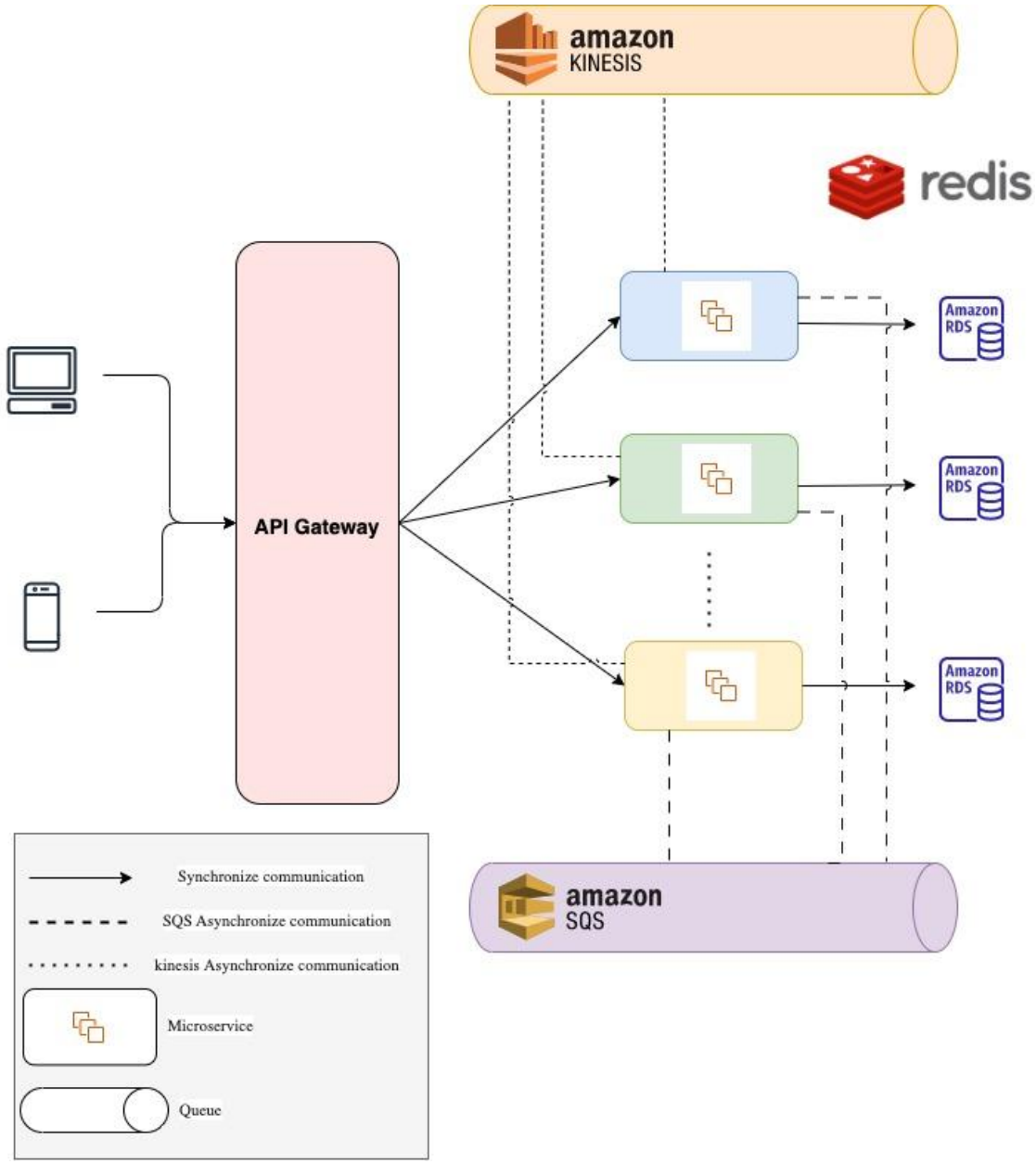
2. System Overview

A fleet of many microservices, all accessible behind an API Gateway, deployed on AWS public cloud.

These microservices are built mainly in Java or Kotlin, most of them using Spring Boot as the main framework.

RDS or Redis for data storage, and AWS proprietary queues systems (SQS, Kinesis) to

exchange data between the services.



3. Assumptions

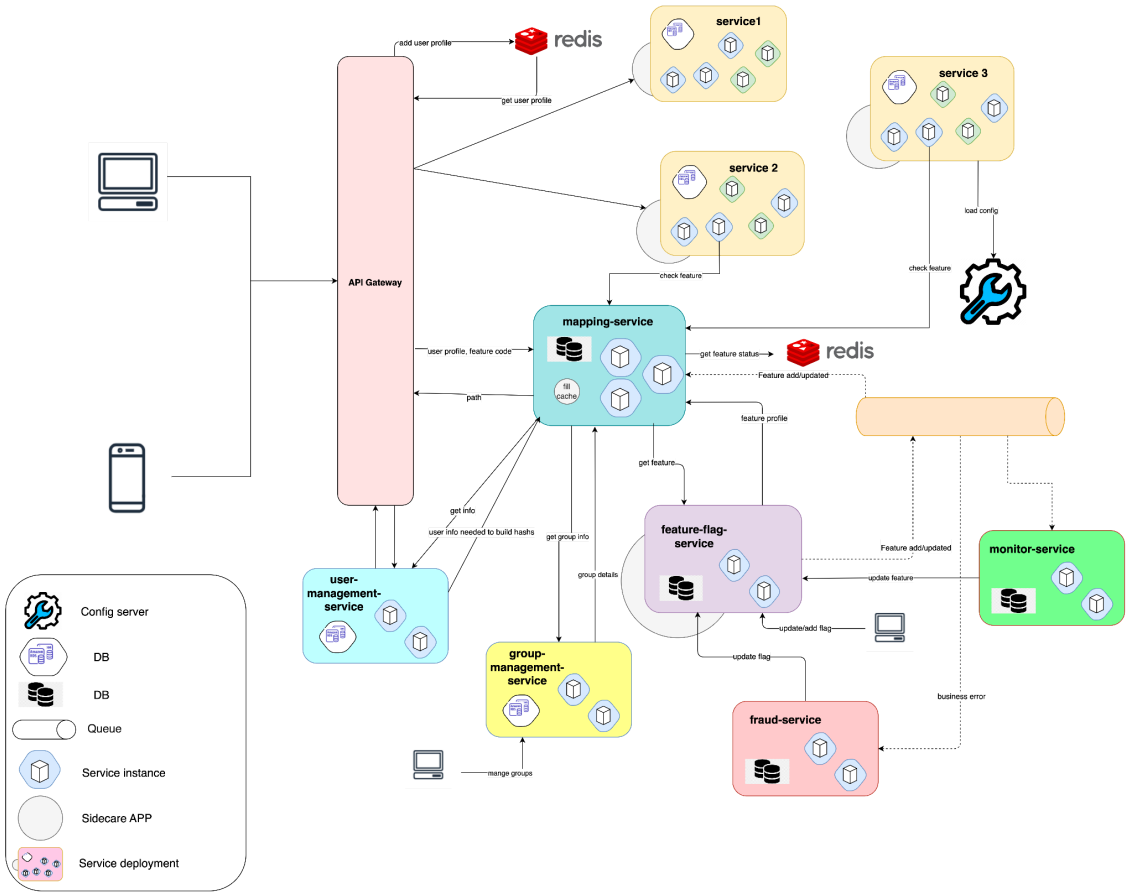
- Add group concept as definition of set of users grouped for business need
- User can exist in multiple groups
- Feature defined by unique code in whole system

- Feature’s rules defined by
    - Country
    - Group id (set of users)
    - User id

4. System Architecture

Add mapping service to be used on 2 levels

- Used by API gateway to manage business feature visibility for specific user
- Used by service instance to decide which API version should be used for that user if we have multiple versions from that API



4.1 Deployment process

Features deployed on services gradually by deploying instances with the new feature

4.2 Failure scenarios

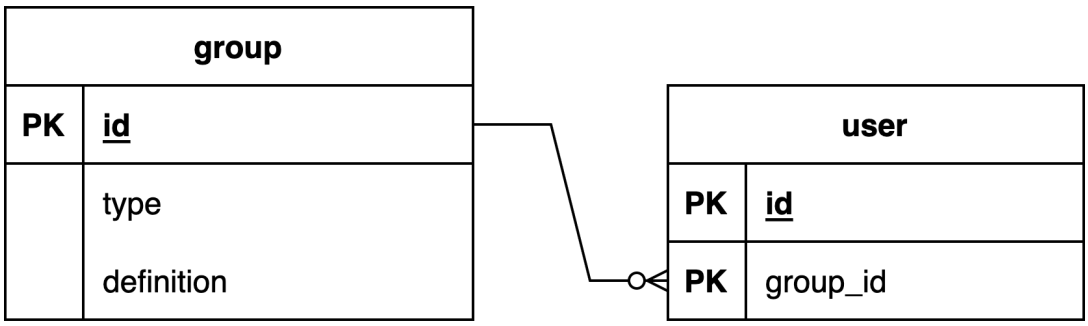
Sidecar application to implement circuit breaker, forward failure calls for alternative paths , set service on read only mode

4.3 User management service

Existing component for user profile management.

4.4 Group management service

Service to manage user group to define set of user linked for business need

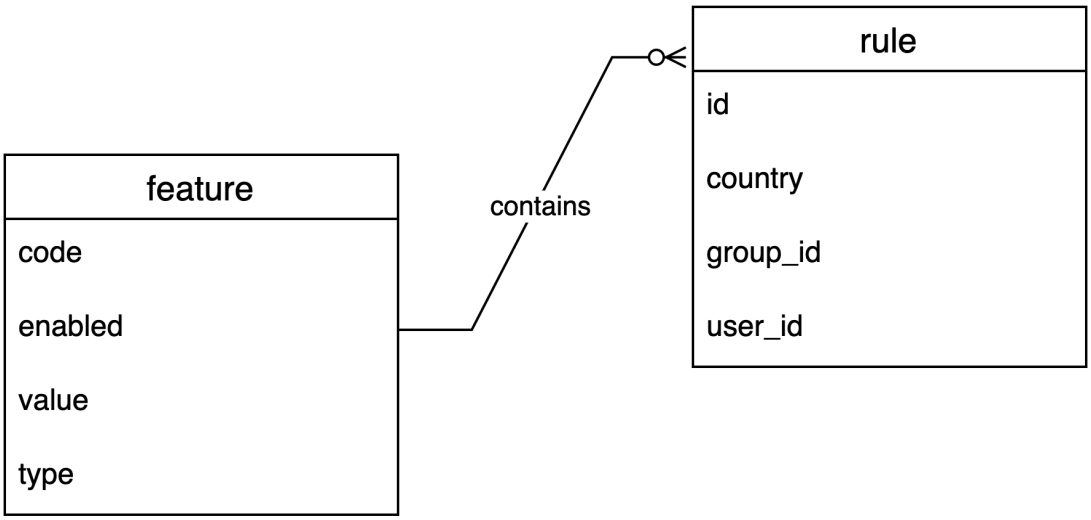


4.5 Feature flag service

Service contain feature data and rules for each feature

- Feature defined by path

- Feature rules defined by
    - Country
    - Group id(set of users)
    - User id
    - Except (group or user or country)
- Multiple instances to avoid single point of failure
- Nosql DB each feature contain all rules that define it
- failure flow should return feature is not available
- Feature enabled/ disabled manually by admin, or by other service Ex. fraud service , monitor service
- Feature type
    - Release flag →
        - non completed feature as disabled (Temp),
        - define rules for beta version testing (Temp),
    - Experimental flag → used for feature versions (Temp)
    - Operational flag → heavy feature that can be disabled in low performance cases , fraud flows (Permanent)
    - Front end features → out of scope



4.6 Monitoring service

Service to monitor system problems

- Non response service
- Long response time

And based on that calling feature flag service to enable or disable feature flag or update feature rules

4.7 Fraud service

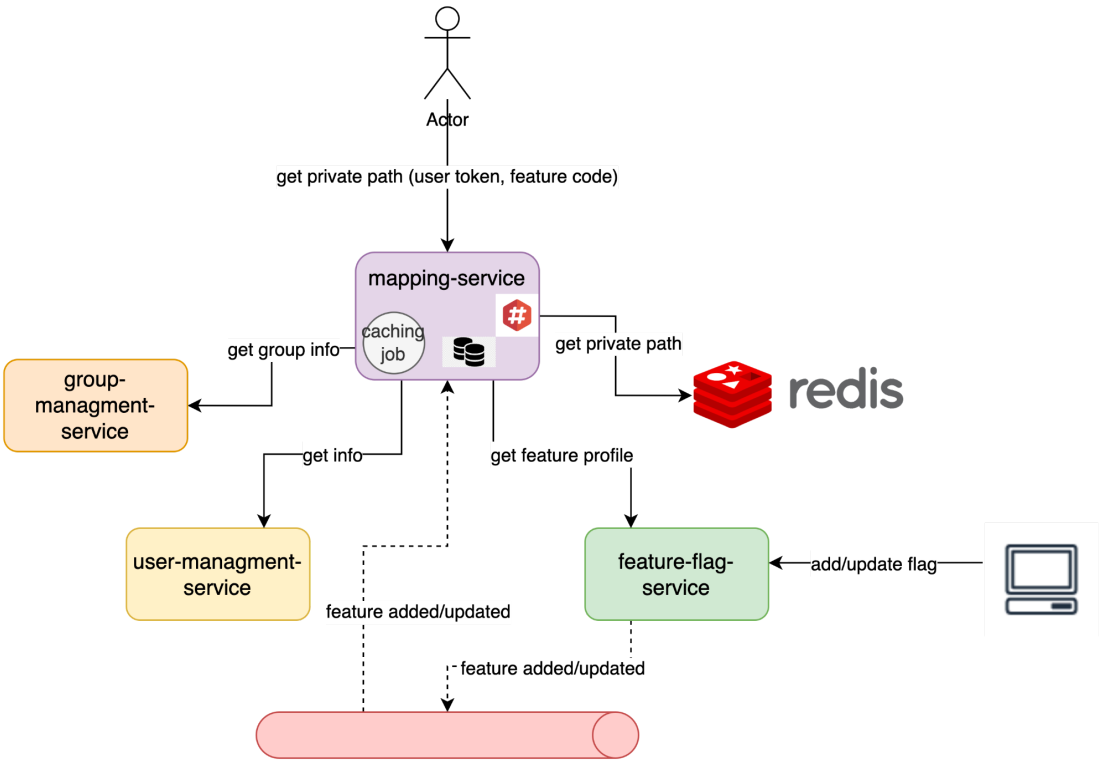
Service to monitor predefined business errors that categorise as fraud error

Ex..

- Repeated failure in payments flow
- Trying to do action on another user data

And based on that calling feature flag service to enable or disable feature flag or update feature rules

4.8 Mapping service



Multiple instances to avoid single point of failure

DB

Feature usage date history to be used in caching job

Caching job run history

Hashing

Hashing mechanism that generate hash value from user profile data (user id , country code , group id) combination

Caching

Use redis to cache (active group or customer hashing , feature) for enabled features and value should be the path that will be used

Cache key: featureCode: userProfileHash

feature with key featureCode:* → valid for any user

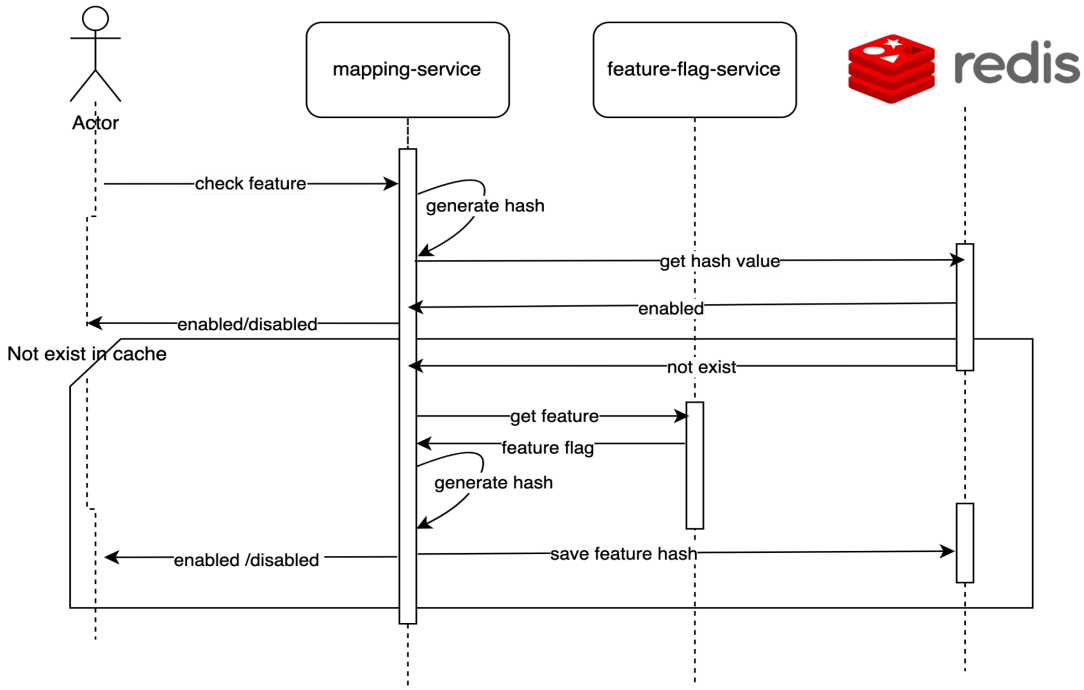
Caching job

Job run periodically to update caching values based on service usage and active customers

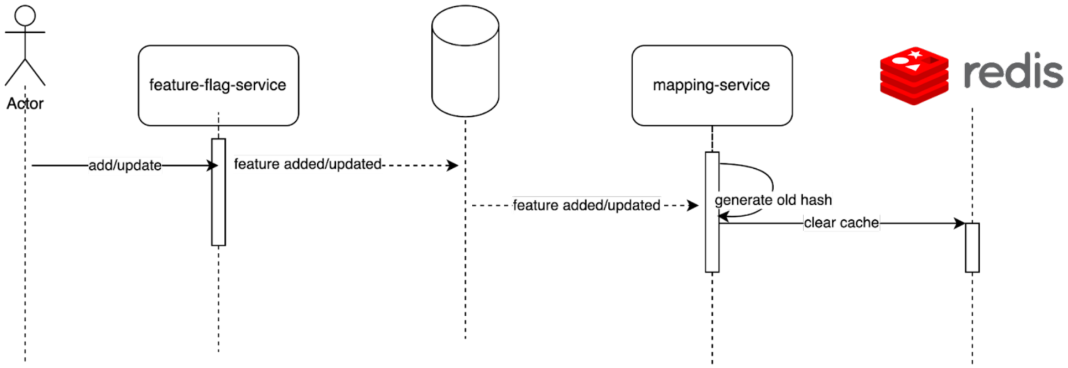
It should run in low load time (details out of this is phase scope)

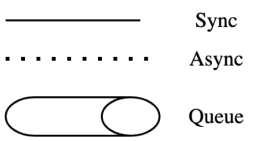
Main Flows

- Check service feasibility



- Feature flag added/ updated





5. Trade off

In mapping service caching layer added to enhance response time, but we cannot cache all system user feature mapping

Solutions

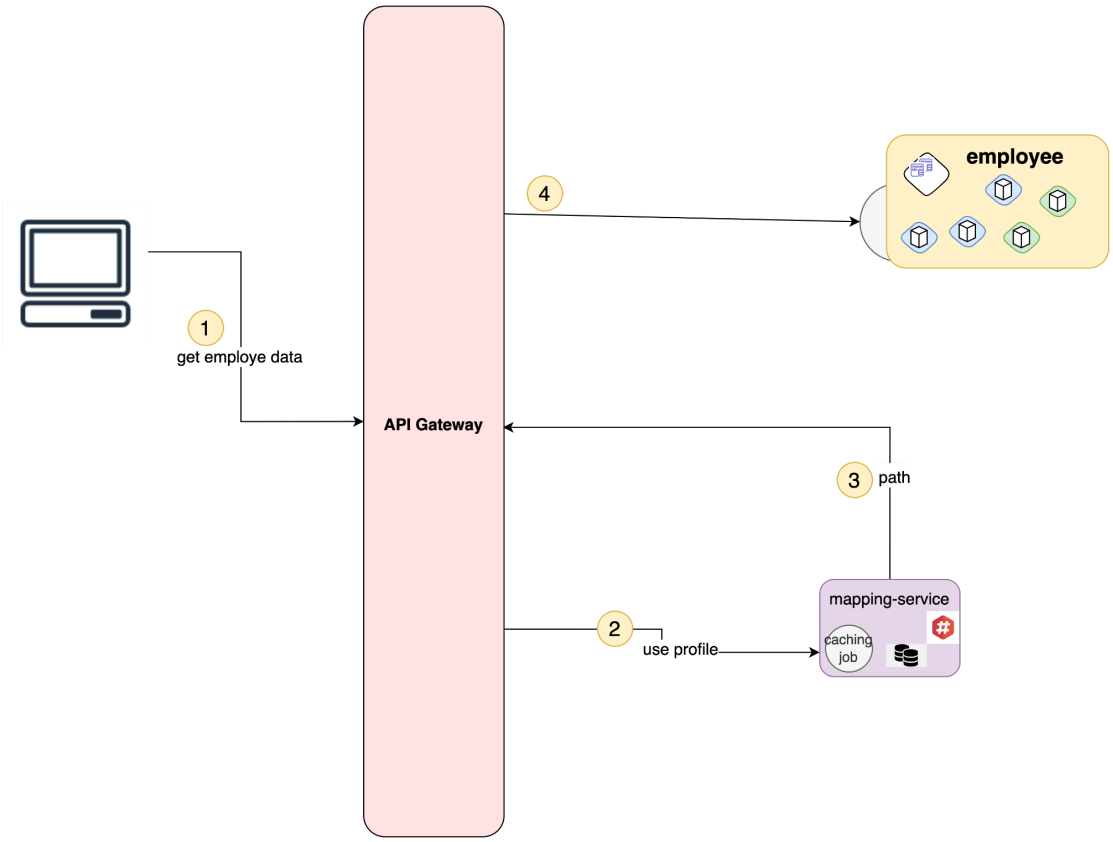
- Make sure to carefully monitor temp features flags and clean them when they are not used
- Prioritise which data will be cached based
    - Feature usage history
    - Active users on the system

6. Out of scope

- Frontend feature toggle handling
- Caching job details and algorithm to prioritise data to cache

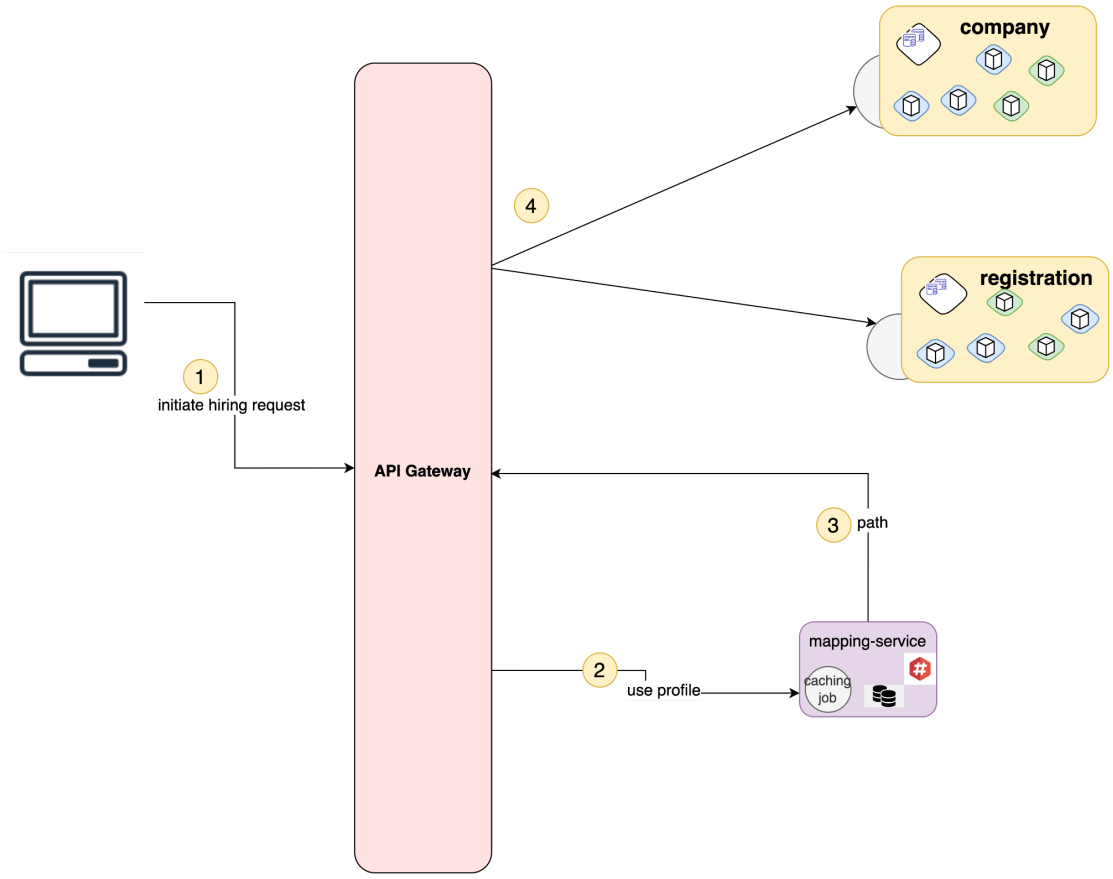
7. Use Case

7.1 get employee profile data request (2 version from get API)



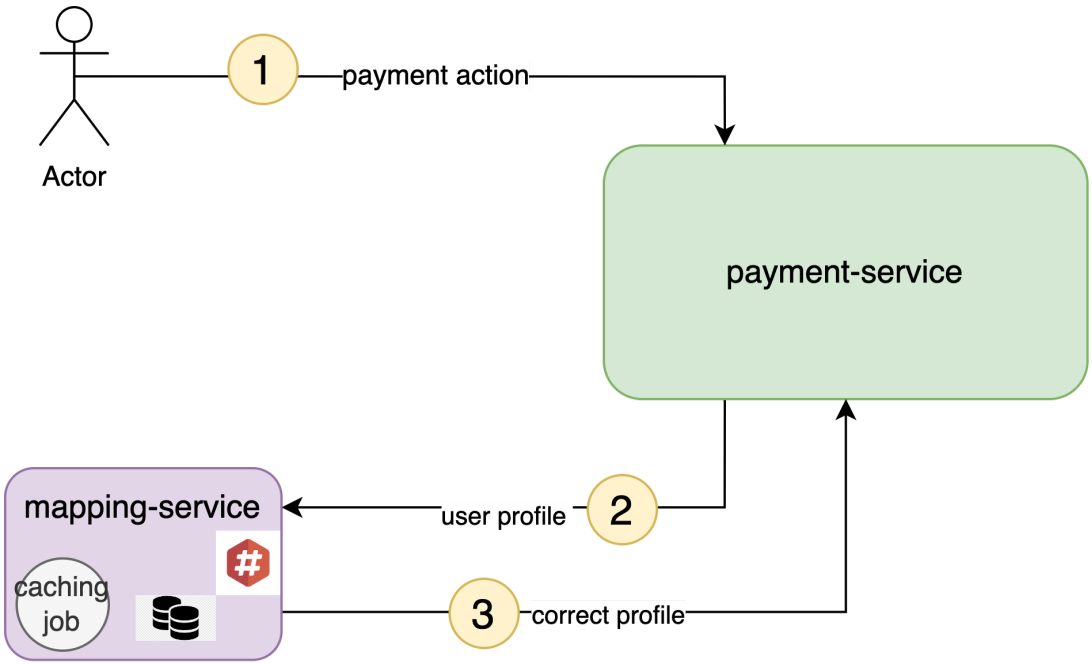
- User request to get employe profile data
- API gateway call mapping service by user profile
- Mapping service check user profile
    - User country configured to use new API that display taxes data → Return path for v2
    - User country not configured to use new API that display taxes data → Return path for v1
- API gateway get the path and call correct API

7.2 initiate hiring request for employee (for anonymous and registered users)



- User request to initiate new hiring request
- API gateway call mapping service by user profile
- Mapping service check user profile
    - anonymous user → Return path for registration request
    - Existing user → return path for get user company details
- API gateway get the path and call it in the correct service

7.3 Disable credit card payment method for specific set of users



- Request to get payment methods
- Payment service find this conditional flow
- Call mapping service by user profile
- Mapping service reply by correct profile for that user
- Payment service use the profile name to select the flow for requested API (return all payment methods except credit card)