Real-Time Collaborative TODO List System Design

Part 1:Summary

Background

We would like to invite you to the next round. The goal of this module is to evaluate whether a candidate can write clean, reusable, and unit-testable code. We expect senior+ engineers and managers to be actively coding and to have a solid understanding of popular design patterns and data structures.

Requirements overview

1、Develop a scalable and well-designed TODO list API application that
2、allows users to manage their TODOs,
3、demonstrating your backend development skills,
4、API design expertise,
5、and software engineering best practices.

We use a relational schema optimized for sharding:

Project Information

Information List	Data
TD version1.0	2025.12.07
TD version2.0	2025.12.09

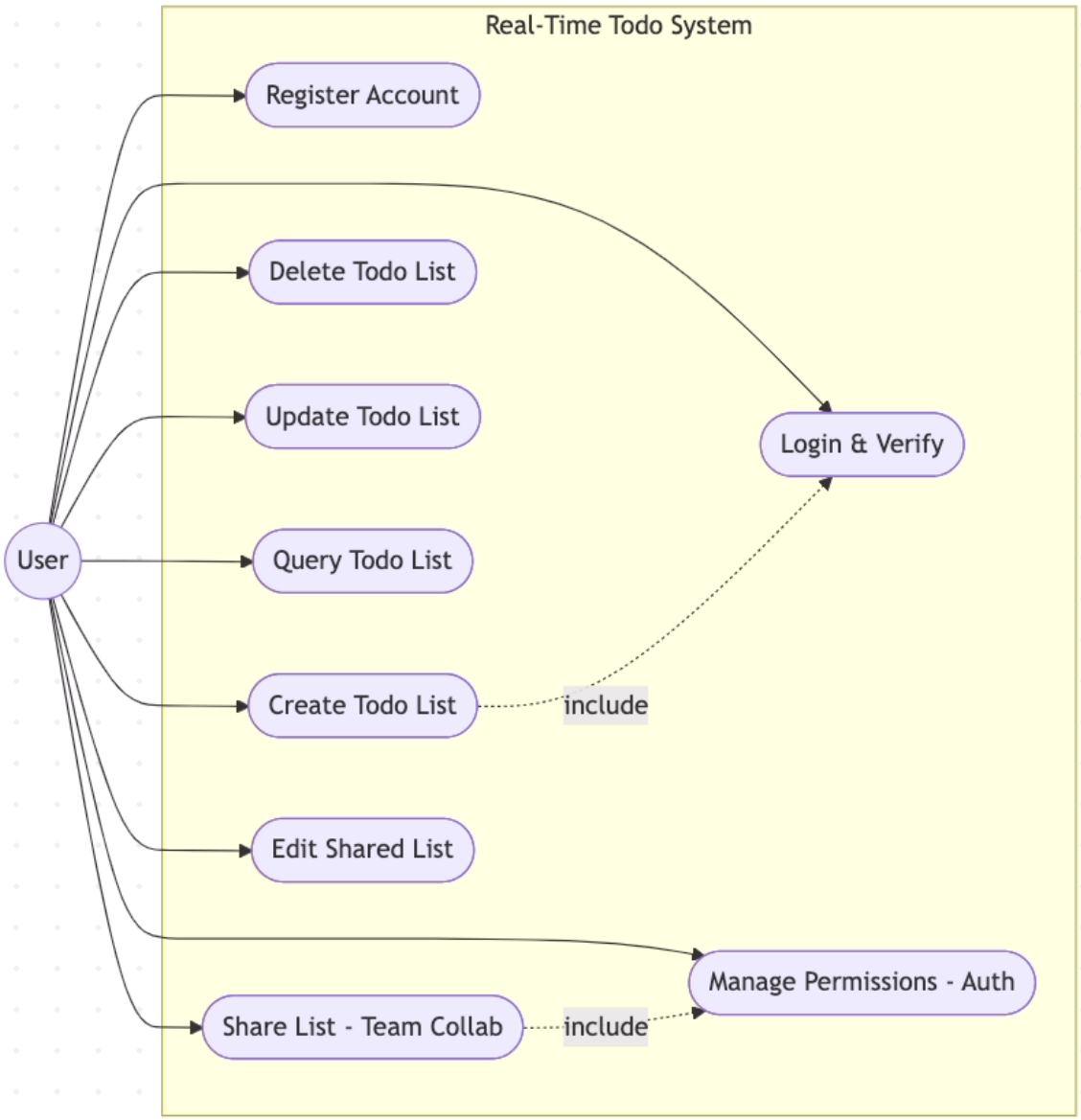
Role	members
PM	
FE	
BE	
QA	

Part 2:Requirements Analysis and Design

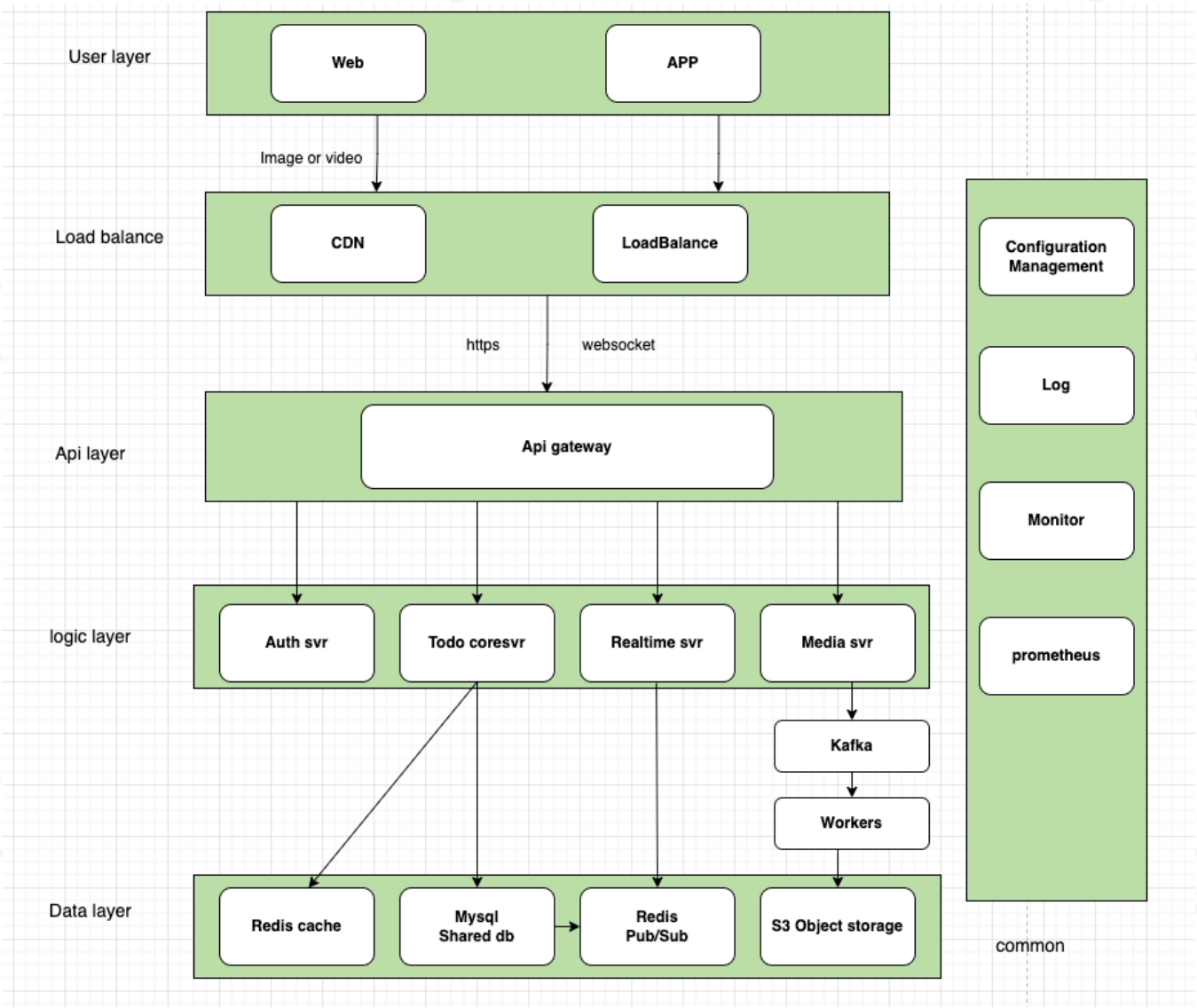
1.Business modedling

1.1 business usecase

https://mermaid.live/edit#pako:eNp1U9uK2zAQ_RUh2OJAHJLYiS8PC9ukD4UstNmkD7WXorUnscCWjGRD02z-fXXxpI2V-MFjH50ZzTnDnHHBS8ApPgrSVmizzRlSz97z9hLEaJQzC8j-xTKeTrKDJsvxFkjt72gDaMdLPuA5frZ8_ZRUQNFRztDu8190v_q1haOXqRdVKQI9FAXvWfc8-sDZ8CNlXmYC-oR-gKCHk8NZCSAdeJmNto-NKurQ1lCDptl4k7ZvS1PNxpu07z2Ik5eZcJP0VBGhSplgzpGPdkAatOJ1TV4c9kPfVV72SBg5AvoGoqFSKtukStJHrmpT4ktJOy_Tb2RuKT-0Aax8n9zdHdpCTfQcZEVbOQwY-f79MAsHMZY7mHXYAa2fDmjdc0BjloOZrh1Mq3Vvvqr9R9AaWqUQWEFBIu8rK-q-BDlov7aL_Il__0rt6asrbejgf5JuAo_VQtASp53oYYwbNROif_FZZ-e4q6CBHKfqs4QD6esuxzm7qLSWsJ-cN--ZgvfHCqcHUkv11xtz1pSoXWquqNBaxEovAU7jxNTA6Rn_xuk8iCbzWThNwkUQL6NouhjjkyLNJ-FsGYTTYLIIlkmYXMb4j7l1OoniMJ7Gi3AWBNE8iVQ5UOZx8Wj33Kz75Q08TTJj

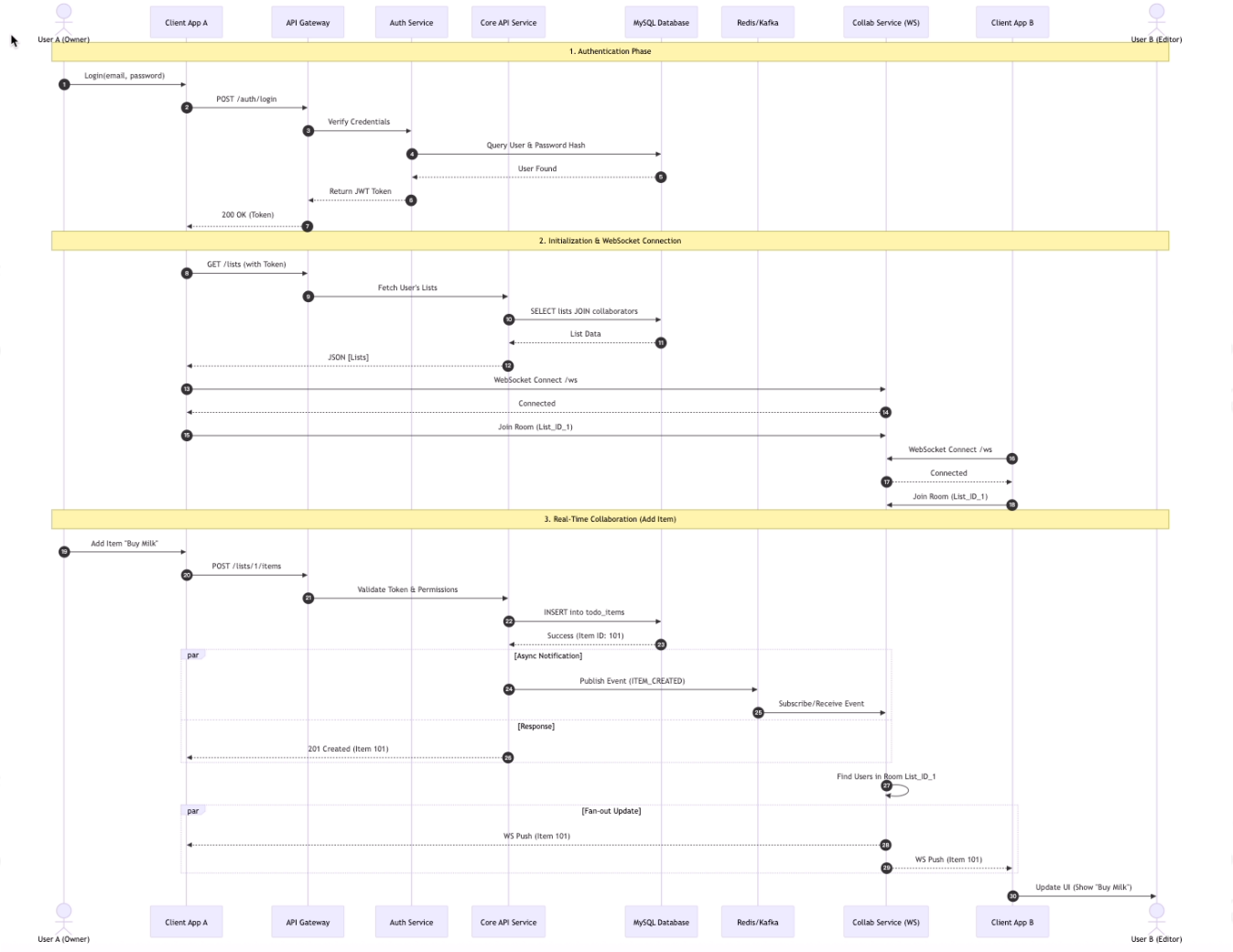


1.2 system boundary



1.3 main shipment flow

https://mermaid.live/view#pako:eNqdVmFv4jgQ_SujfNhLJQoJLV3Ih5UI0L1028ISupXuOFUmMcUqxJztlGOr_vcb2wECBe3p-EISP8-befM8yZuT8JQ6gSPp3znNEtpl5FmQxTgD_JFc8SxfTKiw98XTRHEBD5KKNhBpLqANbn-VUXFmIUsiFEvYkmQKOnNGM2Wg9hLayyW0PwK_EkVXZK2B7UG0uf2Ia-dqZkD6P6bilSX0CApDaEouqLk-CeyGGne3jr_fQpcoMiHyCCoU_AXrROSQpkzWvpHpCzlSLJ_PycQSm6uCFtzH-KQ24YE24aHO4VbnENxeyvA5BrOoe64o8FdadKRiNwTgV41AGJMlRDGewWC2rcxAz798KXoTwC1_ZpILF4TNK5ihlCsu0iLhAoTwoiUBDPrxCGpoj1ltrndaYLGMQM0cwA8q2HQNHUFTnQaZS4vTqwjqYpbfcyrWtrRPMCh44XciZxbaDc-34QzqmudZWgpTzmpIVS4yuHkcwQi7dZBVudy650H_G7gG9ksp61WIMqYLYD-tlJ_gkU5inrxQ3fIso4l-fFKurz1Ua86kkuCuGLp2w3ug2iAK4JqqZGaYf5Nwq_cU1Q6iQrO4d9vrjMDGu-lH95AYr3FB0BhyTzgdUQcxzt4FKmtxE_fv4U_D9Ff5mO_qsFYOPtYMtVVBZyF7cQsMTU-Fu-EsgyHnC3A1-1PUffK3vShOxv9jD0-wh_-V_ZQTLqpoMjI_H7EFLVi16toSbjtNIVJ0cXbiiG3WYeyE-Rru2Pxl7PzihJke1_waw33yqF1-oClTfGQtpQ8RFQsmJab0wTjRfdwbjoBlioPiKX8qhS35Jc6ThEq0qsk26uIs8fyzsjVwhEFbrrNEC8WmxYCxaztKOzKxlHyCdcyg96oHnBuNendPnWGvPep1z3Z7LLrUoDifyESwCa0NaULZK7UBitmYpdgJucQq6T7v_jH39fRBedKiml0lVI-RPfvsqK8ZhtcNILAxydYjhzpck-yc5woelroNu1yOnIjHGLVAIQ5TOWHhU_By5ltbFwa1WcBDBG4846s9r-F2p-I8C5Y6gRI5rTgL9ArRt86bjjd28H2xoGMnwMuUTkk-V9qi77gNX1d_oA6bnYLnzzMnmOJMx7vcsBafDlsI5klFB-e1coK6d2liOMGb848TXF741Yurluf7XrPVaOrFtRP4n6-qrVYTFxpXrVajdfVecX4aUq_aal763oX32bus1xvNRsWh5jV4Z79ezEfM-7950MaZ

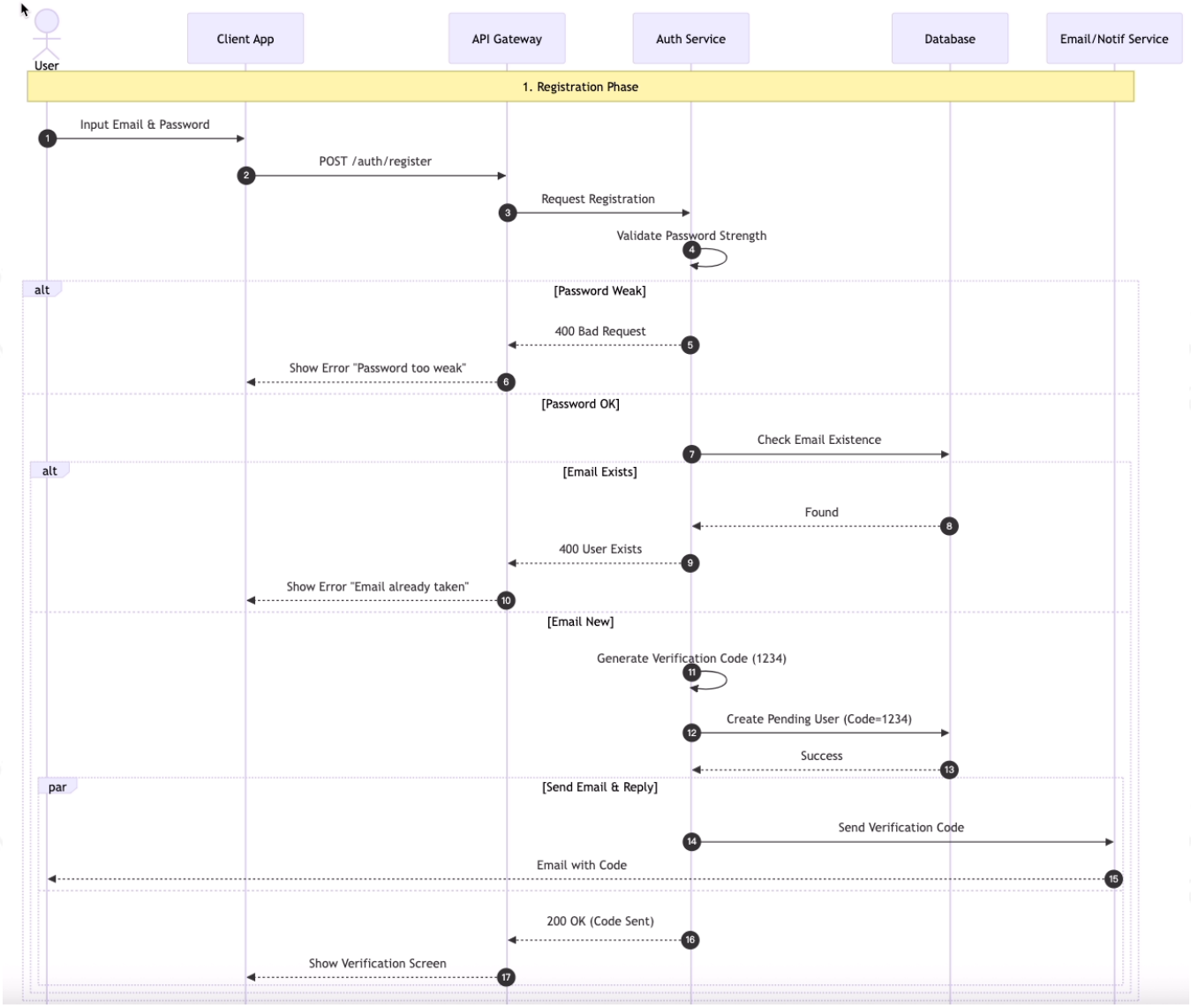


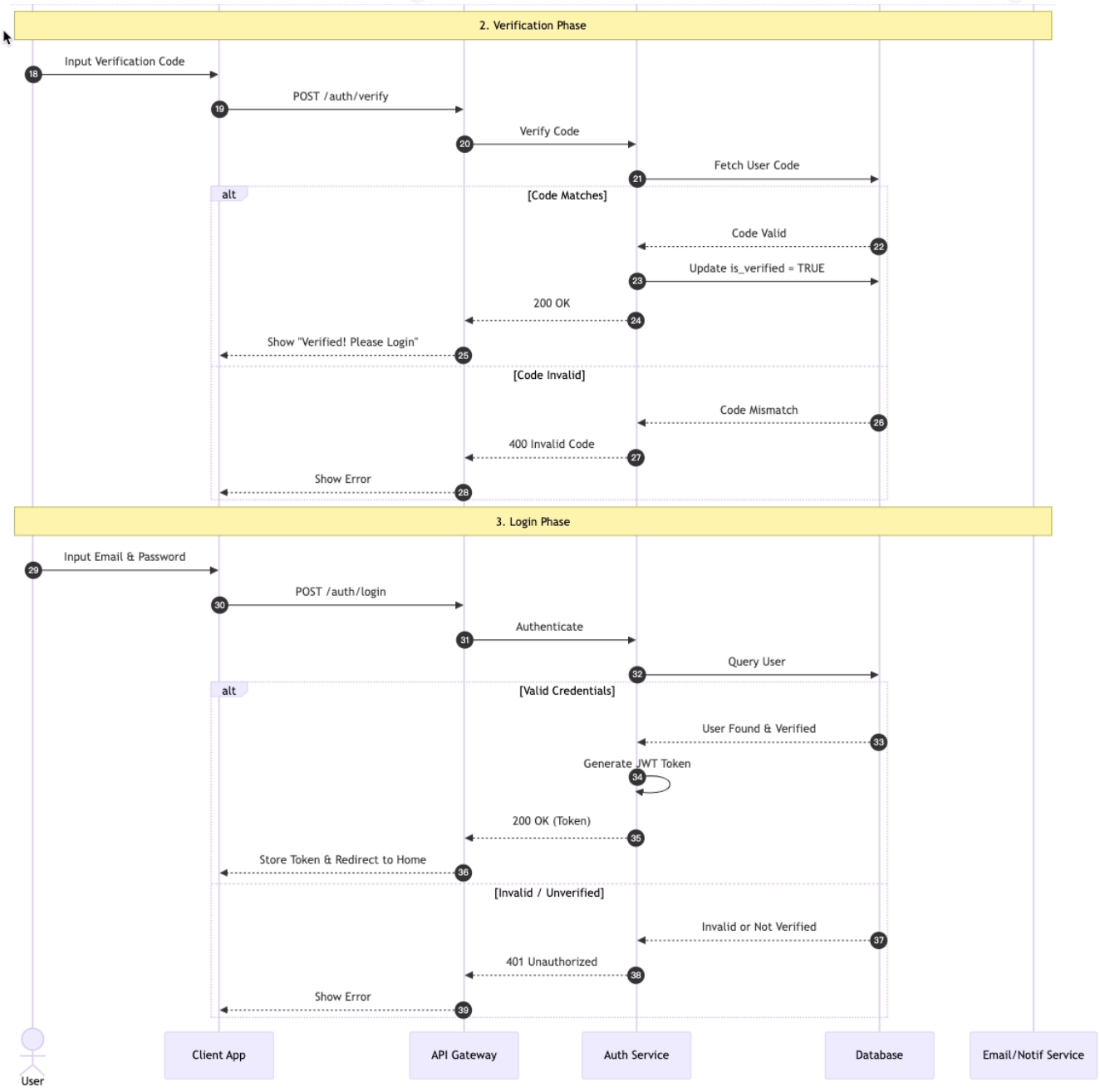
The query flows will be shown in the detailed usecase sequence, refer to 1.3.4.

1.4 business usecase detailed flow

1.3.1 Register Account

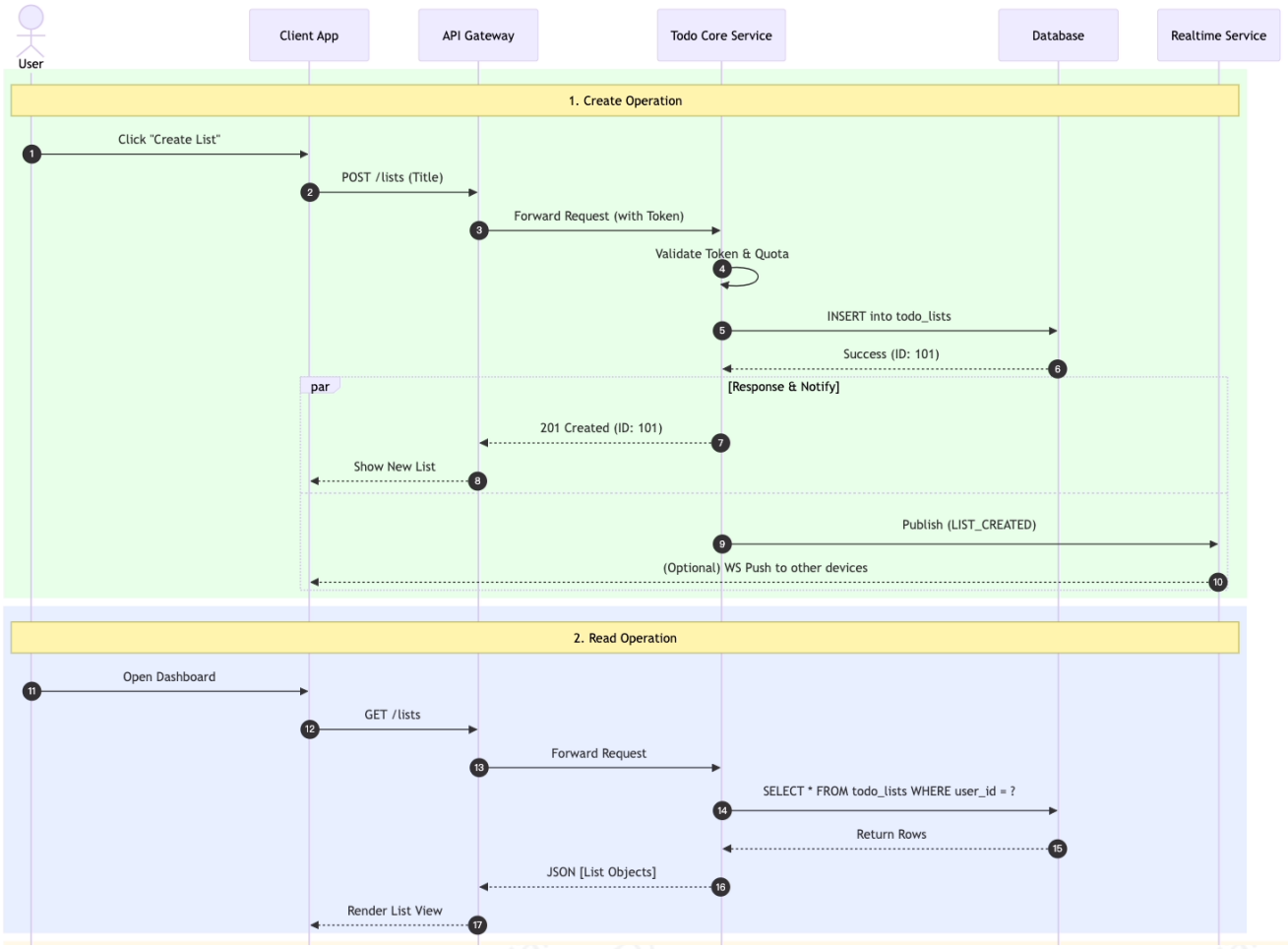
https://mermaid.live/view#pako:eNq1Vm1P2zoU_iu-_jANqStpS98iDQko2-XuAh0FJl0hXXnJobVI7c5x2hXEf9-xnaRJk5Z9WT60cfKc1-fxiV9oIEOgPo3hRwIigBFnU8XmD4LgxRItRTL_Dsqt06eBlorcxdnTBVOaB3zBhCZnEQf8Y3F2d7JYVFGfmYYVWxvYyfgiW1ZxJ4meWZD5n4Ba8gCqqNGpwYyYZt9ZXPP-fM54ZCD25vBKav648ebwQmogcgmuroaD-qTVJDcw5bFWTHMpyHiWRzC4D8fHrkyfXIhFkoV6R8YsjIdShQ7qMAhOC_XJ-HpySw6xvbNDZf1nvUwRiDU1-xgdaYl1KYsiFwaVg-9ZxEN0kIcnE61ATPWsRF-kN4BvwJ7c49xbMc8jzyOnLMzS2CCzPAsdmMzkipwrhdp4oHkALSVZmSDUGUMUF_K7_rIV_Ph4dOqTsxkET2kzz3-a9oiM-E0dWS0FWLx5Y67R6Ye8N59kIsLy69pqDa21zt6qOFVZpICFa6LZE4is5rxuh7mCVU0iWaKfQYAyJN6D4o88cMI7w11K3rfanaODWlvbNQxtyAcRcjF1lbw3hh9r7Iq9mSRBAPFWveUV7ifcMiLMFX4Di2hdxhTSSbePtajUUbWycJOPydlPY6w4bvoqnm3TWEtlG6m8_uLKN2nog6rRLkZLCU8CBSDKxlBMIV_Ym73TpN0s-35rmuzo3N5xsjQ26_phYv2tC54K4vkEOpg5zWzebwaG7eMlQwwUdFIUkUXYCVS7pe8WdjLx-H-bIYeQfCS3N3fne6aPI_HtofNA71Off5FxBNhU8q-cclGaOTa_C7EsZ1ip4JLHc1PmG0MxdbSlz70z4ndV0mm67P_ExyaybakVh_lFQyO3qjq-JqDWhU_-Rhj3rg0KQmPMoh3qsMKyMxgTztiqCGVrAv7z7Zbcyqfi9tu50S3uYD8VeGwB59AOsJArCDR-ocjfcg4FqWTkHpI7sazkWiwrQ-IXAI8Vuwori6eFXg0ZUvHnIvR3xUMbdKp4SH2tEmjQOSjUAS7piwE9UKQRq6E-3obwyJJIm33wimZ4GvpPynlmqWQynVH_EUnDVWI3aHr8yyEYEdQZ8qap3x62rQ_qv9Cf1O_1B83hwGt73f5g0O30hg26pn5n0Ox22sNevzvwPK_XeW3QZxvTaw47R61e66jvtYfdfrc3aFCkADm5dCdQexB9_QXjYDY0

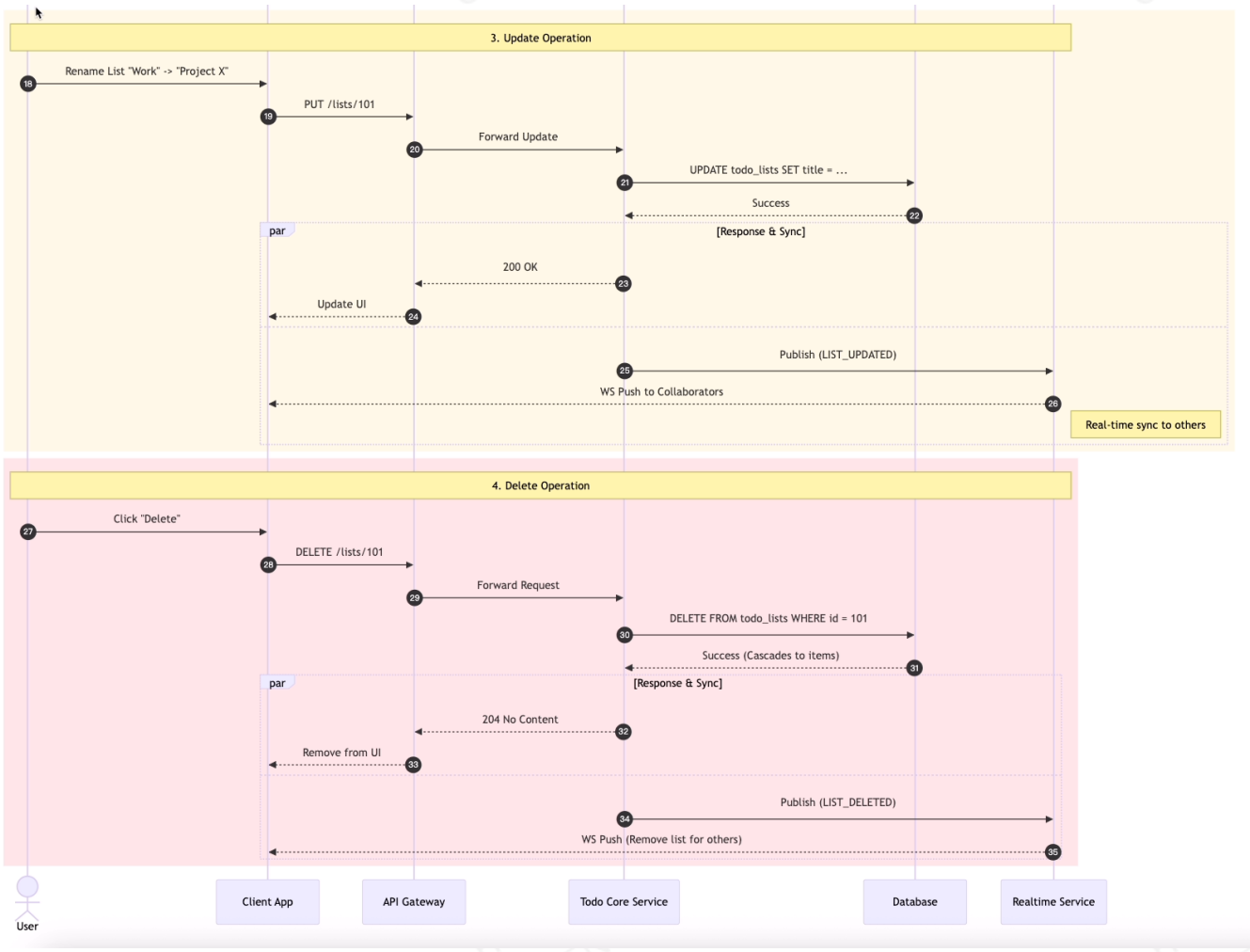




1.3.1.1 Todo list CURD

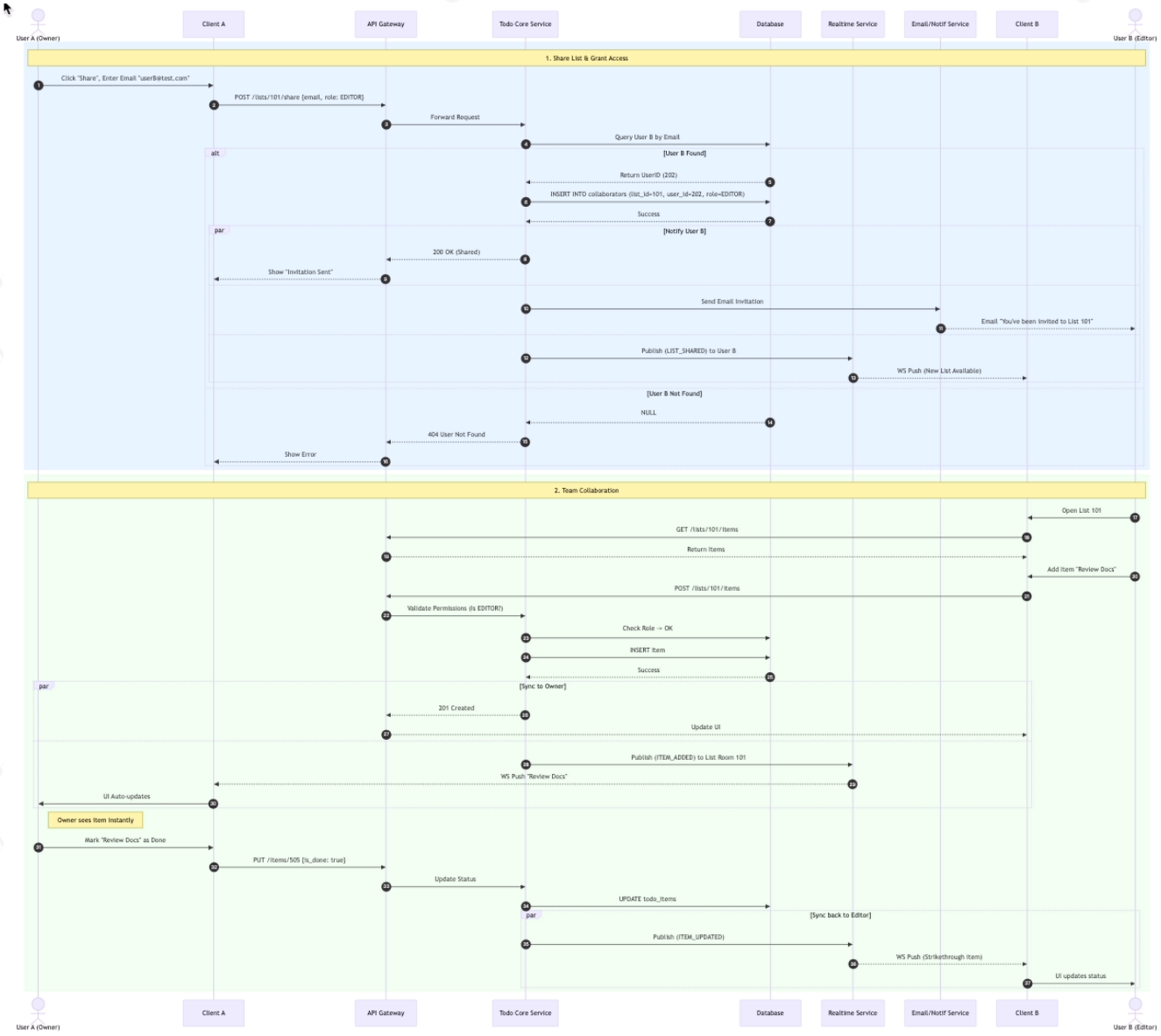
https://mermaid.live/view#pako:eNq1V39z2kYQ_So7N5MM7mCFnwJrJnRsRFJa2xAJ4k5Lx3NIh7lY0tHTKYR6_N27JwlQYgrEcfUH6KS3t6u3795JD8QTPiMWidnfCYs8ZnN6J2k4iQAPmigRJeGUyWycX_WUkDCO11cXVCru8QWNFHQDzvCPxuuz88XiKeo9VWxJVxp2PuyvhztmE5Jp0Ej4Ihu4TH7mHnsKtS800KaKTmm8474z0vcdRgPFw8I8GfLVK3h79LEJqRrQlQyLh9K578MljxUgM33FwpNnzyuZp0DeTUu1eqUMtWYTf-qVfD59RAITis8sa0EZn8wqVDJYMEkVF9E2QMNOO52sIZZujHcPE5IH6KonZIvOYIjPu2LBcOCO4E2AuBhKI64CVqgmR-n5sT8WvBNySaWPTKOekI_Skqs5NvCeRYUojd2EfKQB93UpKQpew4dEKPoEbF9Y0L92e9hJHikBCkVxm1a1hdoXp5tp3cTzWIwl920kqFItpN-eoUSw1Hghophh5muh-Gy1vb1JX-SjVqnmbPu7Ji-yUqDdnYslXLNIRvgGTSN_R7pOR3d1mEzx-eZQuuy7o9uu0zsf9exvMjmjYpLSYKF7T4MTuHExHoORKaHmqBafacUXyGLr1OnJjyyEmqFXFrLxIWFy9WLab2QL4ID218kPKx8REVpEPJ8KlOg-yb_vrRV_tNJ36tXtXfa6I_gJ3jmDq4Ji4eaXntODBOu75T68hZ93a9hhKpEROGIZfzN_sdpf3cE1_Jnaz2D6CTmM_3padoEHBxuOFKYBHzlbvpAK6gaMF-lCLmEGii6LZnhF5T3YImIvoYrUDJuVYxxxW8xhXeTVpnxMyI2Q9xMCpx08H0qh-YTfDzjkeC2XN-gFhyWTVbZTMeOhjau8KBUXtai066JMDMPYa3ZHWJy7iryDBleBwW8HHS3nd9x_jptlz7nfzQoW1hVBQKcCGylk_HVM2n3J7-a4-c7S3utd_jTd5mN82I0D_i_W1zDAZgFbd_PH5V0_St6btN-x4ed17lWyjYaF8vseMe_zv3y63e6XGt9XKXZu310ae9Rnse4jx1er-OSlVN7A_R6vRwpJOKh2h4XYA5hJET5T8hkZx0m-lKfTfMEMfTRT8Ml_SZiUyZ3kPrGUTFiZhEyGVA_JgwZNCEaH2HoLT302o0mQvvM9Yhi-Hf8hRLiOlCK5mxNrRoMYR0m6xPNvgg0k3T66IokUserNdApiPZAvOGo3DLNtNmvVimlWWo1qmayIVTVrRqNeaZ7VWmarZprV9mOZ_JMmrRrNeqt11qjW6-1aq31mmmXCfI6r_Cr7Lkk_Tx7_BUpkl-8


User
Client App
API Gateway
Todo Core Service
Database
Realtime Service
1. Create Operation
1 Click "Create List"
2 POST /lists (Title)
3 Forward Request (with Token)
4 Validate Token & Quota
5 INSERT into todo_lists
6 Success (ID: 101)
par
[Response & Notify]
7 201 Created (ID: 101)
8 Show New List
9 Publish (LIST_CREATED)
10 (Optional) WS Push to other devices
2. Read Operation
11 Open Dashboard
12 GET /lists
13 Forward Request
14 SELECT * FROM todo_lists WHERE user_id = ?
15 Return Rows
16 JSON [List Objects]
17 Render List View



1.3.1.2 Share List - Team Collab

http://mermaid.live/view#pako:eNqtV21v6jYU_itWpLuBRGnIBUojtVsg3C5aWzoSNm2qVJnELdZN4s526LhV__uOnYSE13Yv-QCxfc7xeXnOY-fVCFlEDNsQ5M-MpCFxKX7iOLlPETw4kyzNkjnh-biYDSXjaPKSEo6wQDMB_w5q6IlmLvKMuaQhfcapRKOYklQ6SjR_Rc6u0BWW5AWvlJBz55XDPcYYJ0ooYBHLBz7hSxqSXVF3qARdLPEciz3r00CtTwmOJU2O2Lllkj4q0XGCaXyaDw9K5yEOa9EO61kbR1T9lWkbokY-A3nLxT59QhcfftYqnTbyFxiy8R1yMrlgnH4jqKGmaPqEMLqmQjb_9QachBLxp3nD-my2kNXtwU-vV9hTT8okQWxJClS0ijDtyi_lADh3xVWOnDAkQlTqWunk8rJAiq0yF35F94bWvTfAXirBuC4ATGeQuuGPkgjZDllyb1SWCgtgq0CQje4mfoBOY9hfnHbMzqnQ_rwSZauFOIuJjcauF0ymb5WdQlv5BBiz0RfGXzCPAC3QJUJWgrWtQRDk3aGNfskIX5UFnq9yvytJAFy5-IVlaVStqMcdnqy3nRKZ8VQLey5qWKbV3BSuberd-mOAtHcbTFDI4hjPGcdQBIEaKvgHGl1A-C2kkqcGYCwP_yKPvnnYDT_bKthm5EUL5I1Sxr25vPa1XhnLNNHk5xymJGruapRVqEPDX7AXgICXLqnEkrIUWjGVdQzoFG9ntZYs7aat1KICUZWxXSUtrTwoIV2C8HeWfb8kaE5IiqgyQCIkWQ50SPQ_8WgaAEyzOZRpgRrXnh88-D8507HbVAYPpXMaVHmB8v_mgwmlf0teciecJTiK5zHZyiype0JiQUowQqjvAfJ2dn29B4H1qnbNbm7wgLmDRR1zznjNs1JRv_wXcrTamuVPAkXzo3VrKOw0RiwNM84VT6sCA1v-HzTZNTVDKq40P0KT4GFAcLLp3D6SAa_KEySKBIAKAe6Saj1frONi8gz4XINyiyiH9cJdjTd4UhkWu4xYt13Qk7cpecwbJ4q0OPTPlCwpQNVlodhl8OExBj_gWYnRX3FMI5hDd4QnVAhIJXCgJwqW_6G5l7NHCwJHzhT4EJ1cAi3tFSo5diPnR5myeIMM6a_SUPW0Lv87fWSZHTTiBIbvNRA4NnvWEc-82imz3Xj7yMYLxjcPjusWXKNxMmUs2QTLNt04Fd0crOPmaawDBjc9dT1hJ5l2V-xp9h2459fMBPOvArCOIpaSrYbi9GkhEXtExSa5hiBE6OYAchYSbh3x6siF4wbsb8eib48b2-27XcwAmhqQpz2zh16peFAu2kjyjBy5UBQF8-HYycRerM3uXCcYQ1Ui9kAPddgaVHMM6IUK5i33wcLnW7jNw5WuHyy-5PQrkQvOsqeFTm1zX7mHtbMS6l2UGomtSDfp3WgZT5xGhq3S1jIS6FyshsarEro35IIkcBG04TUijziL9Yn_Bmpw6f4DAFtqaucM-xHDudYy8s2LT5q1COxI-AhOJmnY3V5X2zDsV-Mvw-5ZZtvqdc8H_f55twN_LWNl2Fa3fd4fmFanZ_VNa3BmvbWMb3pTsz047_Q7gzPzc_9s0B_0zlsG0cHf5J9V-uvq7W__HNwD



1.4 Api

1.4.1 Todo list app apis

api def	Request	Response	function

/api/todolist/login	{
	"email": "user@example.com",
	"password": "*********"
	}
		{
		"code": 200,
		"message": "Login successful",
		"data": {
		"token": "*********...",
		"user": {
		"id": 101,
		"email": "user@example.com"
		}
		}
		}
			Authenticate user and issue Token.

/api/todolist/register	{
	"email": "user@example.com",
	"password": "*********!"
	}
		{
		"code": 200,
		"message": "Verification code sent.",
		"data": {
		"is_demo": true,
		"demo_code": "1234"
		}
			Register a new user account.

/api/todolist/verify	{
	"email": "user@example.com",
	"code": "1234"
	}
		{
		"code": 200,
		"message": "Account verified. Please login."
		}
			Verify email with the code sent.

/api/todolist/create	{
	"title": "My New Project"
	}
		{
		"code": 200,
		"data": {
		"id": 1001,
		"title": "My New Project",
		"created_at": "2023-10-27T10:00:00Z"
		}
		}
			Create a new Todo List.

/api/todolist/read	* *Headers*: `Authorization: Bearer <token>`
	* *Query Param (Optional)*: `?id=1001`
		{
		"code": 200,
		"data": [
		{
		"id": 1001,
		"title": "My New Project",
		"role": "OWNER"
		},
		{
		"id": 1002,
		"title": "Team Shared List",
		"role": "EDITOR"
		}
		]
		}
			Get all lists (or a specific list details).

/api/todolist/update	{
	"id": 1001,
	"title": "Renamed Project"
	}
		{
		"code": 200,
		"data": {
		"id": 1001,
		"title": "Renamed Project"
		}
		}
			Update list metadata (e.g., Title).

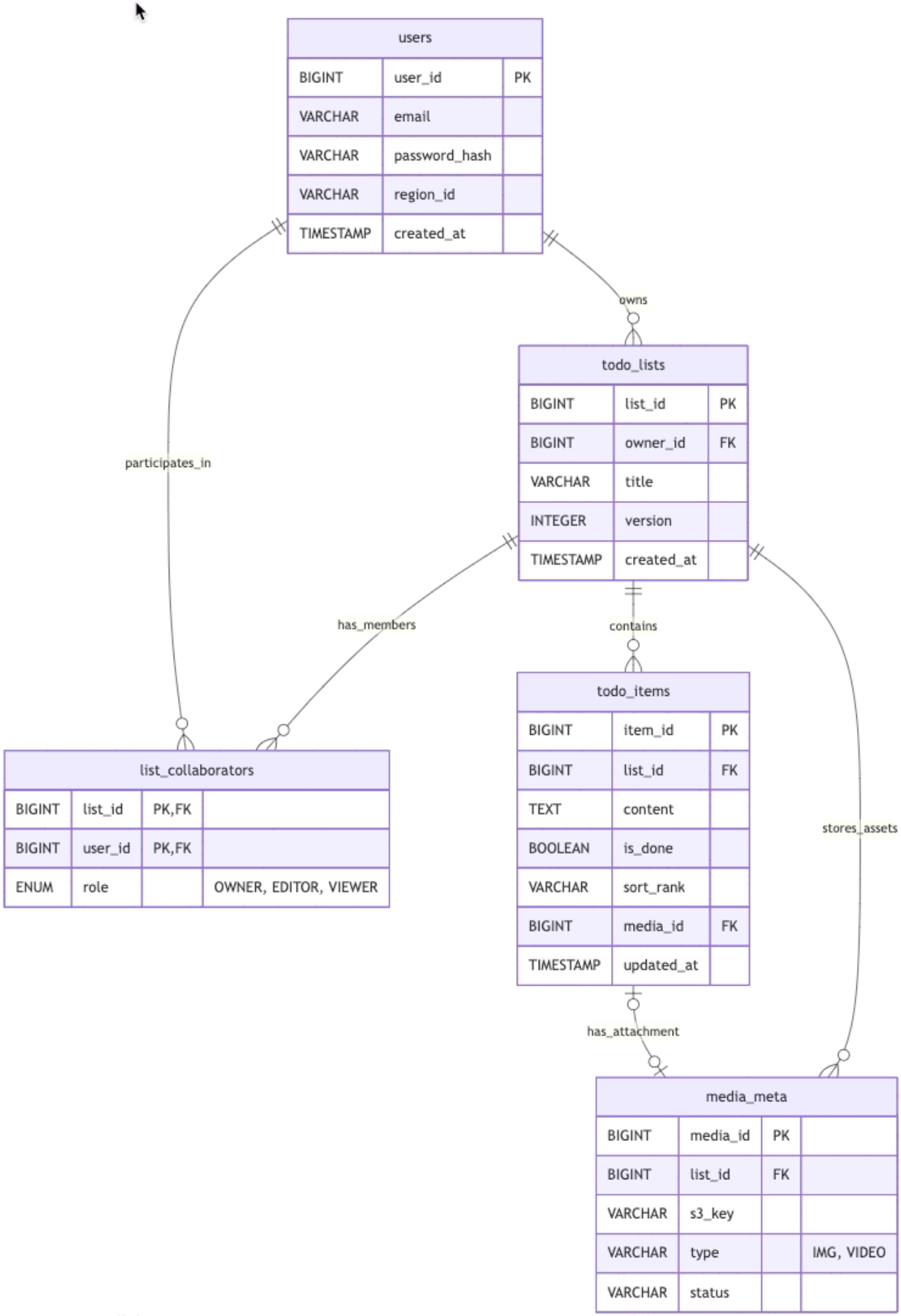
/api/todolist/delete	{
	"id": 1001
	}
		{
		"code": 200,
		"message": "List deleted successfully"
		}
			Delete a list.

/api/todolist /sharelist	{
	"list_id": 1001,
	"target_email": "colleague@example.com",
	"permission": "EDITOR"
	}
		{
		"code": 200,
		"message": "Invitation sent to colleague@example.com"
		}
			Share a list with another user.

2. Data model

We use a relational schema optimized for sharding:

https://mermaid.live/edit#pako:eNqVVG1vmzAQ_iuWpX6jEQl5Ab5lDcvQlqTKWDtNSMgFN1gFjGyzNkvy33dAk5BCq80Swr7z43vu8fl2OOQRxTamYsbIRpDUzxCMQlIh0a5elOOTO3eXXmUPWIRuv55dd9P1zZfpGtGUsKRtzomUz1xEQUxk3HYLumE8gzPPLs9dON-96eIWhYISRaOAqNp78LN6onjEg4RJ1UWytL8l-eriz1mdwOeOBBRTCT2bYb8zd9boN0gBFP-LX0Uh5ElCHrggiouPeWoXfFpaX7qd5Y8FEjyhyMer-6Wz1pAzc70V_O9c595Z-7hLLaZo2sWitL-j1pFgM7rn_PRQyDNFM9UArFbfnOkSMRlEPKNtbSUXKhAke2oFSWnESCvKSeIij7olrnEpVaQjqdOh_5jViaYRPNFtR2ls81JudzEvRZ45q6PGF2BFVCHf0Ly6QmuaEAUVJGOWy-b72u-vr_muWcx2WaFdezoKyoanJRQLWQ4CyYC1n8ZHWHiNIF76ADFq3HvoRvHY1cUTlrUh9YY9B8i-eTV1IKIUCeP0VDLtKBcQCRwhI2gcVEEorOGNYBG2lSiohlMqoNPAElf37mMV05T62IZpRB9Jkajydg4Ay0n2i_P0iBS82MTYfiSJhFVdWa9t72QVNIuouOFFprA9saozsL3DL9g29EFvMDCsoW5O-iOjP9TwFtuj3sgajibGRNct-EzzoOE_VVC9Z5rmaGjpY33cH1t9Y6xhSBOSW9RNt-q9h7-R6aoi


users
BIGINT user_id PK
VARCHAR email
VARCHAR password_hash
VARCHAR region_id
TIMESTAMP created_at
owns
participates_in
todo_lists
BIGINT list_id PK
BIGINT owner_id FK
VARCHAR title
INTEGER version
TIMESTAMP created_at
has_members
contains
stores_assets
list_collaborators
BIGINT list_id PK,FK
BIGINT user_id PK,FK
ENUM role OWNER, EDITOR, VIEWER
todo_items
BIGINT item_id PK
BIGINT list_id FK
TEXT content
BOOLEAN is_done
VARCHAR sort_rank
BIGINT media_id FK
TIMESTAMP updated_at
has_attachment
media_meta
BIGINT media_id PK
BIGINT list_id FK
VARCHAR s3_key
VARCHAR type IMG, VIDEO
VARCHAR status

2.1 user tab

```
### 2.1 User Data (Meta DB)
*   **Table**: `users`
*   **Sharding Key**: `user_id`
*   **Strategy**: Range Based + Hash.
    *   Since user growth is huge, we can start with **Hash Modulo** (e.g., `user_id % 1024`) over logical shards.
    *   **User Mapping**: Maintain a `uid_mapping` service or table (GID -> Shard ID) if we want flexible
migration, but consistent hashing is simpler for 1B users.
*   **Physical Topology**:
    *   Split into **16 Database Clusters**.
    *   Each cluster holds **64 Logical Tables** (`users_00` to `users_63`).
    *   Total Tables: 1024.

-- ====================================================
-- SHARD: USER_DB_{0..15}
-- TABLE: users_{0000..1023}
-- ====================================================
CREATE TABLE users_0000 (
    user_id BIGINT UNSIGNED NOT NULL COMMENT 'Global Unique ID (Snowflake)',
    email VARCHAR(128) NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    region_id CHAR(2) NOT NULL DEFAULT 'US',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    PRIMARY KEY (user_id),
    UNIQUE KEY uk_email (email)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT='User Shard';
```

2.2 Todo data

```
### 2.2 Todo Data (Transactional Data)
*   **Tables**: `todo_lists`, `todo_items`, `list_collaborators`
*   **Sharding Key**: `list_id` (CRITICAL: All data for one list MUST live on the same shard to allow JOINs and
transactions).
*   **Strategy**: **Hash Modulo** (e.g., `list_id % 4096`).
*   **Physical Topology**:
    *   Start with **32 Database Clusters**.
    *   Each cluster holds **128 Logical Tables**.
    *   Total Tables: 4096.
*   **Why list_id?**
    *   Most operations are "Get List Details" or "Add Item to List".
    *   `list_id` binds all items and collaborators together.
*   **User -> List Mapping**:
    *   We need a way to find "all lists for User A".
    *   **Solution**: An "Index Table" (`user_list_index`) sharded by `user_id`.
    *   This table only stores `(user_id, list_id)` pairs and points to the correct `list_id` shard.
```

```
-- 1. Todo Lists
CREATE TABLE todo_lists_0000 (
    list_id BIGINT UNSIGNED NOT NULL COMMENT 'Global Unique ID',
    owner_id BIGINT UNSIGNED NOT NULL,
    title VARCHAR(255) NOT NULL,
    version INT UNSIGNED DEFAULT 1 COMMENT 'Optimistic Locking',
    is_deleted TINYINT(1) DEFAULT 0,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    PRIMARY KEY (list_id),
    KEY idx_owner (owner_id) -- Local index only useful if we know shard
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- 2. Todo Items (Colocated with List)
CREATE TABLE todo_items_0000 (
    item_id BIGINT UNSIGNED NOT NULL,
    list_id BIGINT UNSIGNED NOT NULL,
    content TEXT,
    is_done TINYINT(1) DEFAULT 0,
    sort_rank VARCHAR(64) NOT NULL DEFAULT '',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    PRIMARY KEY (item_id),
    KEY idx_list (list_id) -- Critical for fetching list items
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- 3. Collaborators (Colocated with List)
CREATE TABLE list_collaborators_0000 (
    id BIGINT UNSIGNED AUTO_INCREMENT,
    list_id BIGINT UNSIGNED NOT NULL,
    user_id BIGINT UNSIGNED NOT NULL,
    role TINYINT UNSIGNED NOT NULL COMMENT '1:Owner, 2:Editor, 3:Viewer',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (id),
    UNIQUE KEY uk_list_user (list_id, user_id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

2.3 index tab

```
-- ==========================================================
-- SHARD: INDEX_DB_{0..15} (Mapping Table)
-- Sharded by user_id -> To find lists for a user
-- ==========================================================
CREATE TABLE user_list_index_0000 (
    user_id BIGINT UNSIGNED NOT NULL,
    list_id BIGINT UNSIGNED NOT NULL,
    role TINYINT UNSIGNED NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (user_id, list_id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;


## 4. Expansion Strategy (Re-sharding)

To go from 1B -> 10B users without downtime:

1.  **Virtual/Logical Shards**: We use 1024 logical tables from day 1 (`users_0000`...`users_1023`).
2.  **Physical Mapping**: Initially, all 1024 tables might reside on 16 physical DB instances (64 tables per
instance).
3.  **Scale Out**: When load increases, we move logical tables to new physical instances.
    *   *Example*: Move `users_0064` to `users_0127` to a new `DB_INSTANCE_17`.
    *   No data changes needed inside tables, just routing config update in the Middleware/Proxy (e.g.,
ShardingSphere, Vitess).
```

2.3 DDL

~~slo_status_matching_tab add booking status column~~

~~new booking status push retry tab or ofg status push retry tab add booking flag column~~

no DDL

2.4 Configure Info

Key	Default value	Note

Part 3:Non-functional feature design

1.Performance

API	cache type	peak QPS	db read	db write	remark

2.Monitor

2.1 Business Monitor

1. booking/init booking/batch_init api monitor
2. business metric monitor like forder/init

2.2 Service Monitor

Num	Service & interface	Metrics & Rules	Monitor Link	Attached Business use case
1	/	监控大盘各项核心指标	监控大盘：https://monitoring.infra.sz.shopee.io/grafana/d/qekSHcZVk/omsfu-wu-jian-kong?orgId=7&var-cid=sg&var-env=live&var-link=All&var-system=All&var-project=oms&var-module=All&var-http_api=All&var-grpc_api=All&var-spex_cmd=All&from=1670774400000&to=1670860799000	
2	/	卡单	运营后台： https://ops.ssc.shopeemobile.com/oms/exceptionManagement/dashboard?_p_=fbs&_c_=SG	

6.3 Container Basics Monitor

Num	Application	Metrics & Rules	Monitor Link

1	parcelqueryset, parcelcommonset, fulfillmentqueryset, fulfillmentcommonset, logisticscommonset, logisticsqueryset, warehouseset parcelflow,parceldata, parcelsaturn,parceladmin, parcelapi,parcelquery, admingateway 等	内存、CPU	grafana监控：https://monitoring.infra.sz.shopee.io/grafana/d/RByJXBMqw/ssc-ctl-k8s-monitor?from=now-3h&orgId=1&refresh=30s&to=now&var-Group=Supply+Chain+Group&var-Group_id=groups%2F10&var-Project_id=projects%2F141&var-chassis_datasouce=monitoring-chassis&var-cid=vn&var-datasource=k8s-general-ctl-live&var-db_instance=All&var-db_role=master&var-env=live&var-group=Supply-Chain-Group&var-master=All&var-module=parcelapi&var-module_for_log=parcelapi&var-mysql=All&var-project=oms&var-sgw_datasource=sto-sgw

7.checklist

refer to DMS checklist

Part 4:Deployment

1.Data Migration Solution

no old data

2.Compatibility Solution

3.Deployment Resource

部署无依赖

功能依赖：
OF/SLS

4. Grayscale Deploy Solution

OMS just provide new feature, the grayscale depends on the caller.

5. Rollback Solution

None

Part 5:Appendix