UML기반 설계 가이드

# 사전준비

## 1. Knowledge에 sample 추가

### 1) 논리 아키텍처 예시

**제목: sample\_논리아키텍처**

| **@startuml**  **!theme mono**  **title 요금제 캐싱 시스템 - 논리 아키텍처**  **' 패키지 정의**  **package "클라이언트 계층" {**  **[모바일 앱] as mobileApp**  **[웹 앱] as webApp**  **}**  **package "게이트웨이 계층" {**  **[Application Gateway] as appGateway**  **note right of appGateway**  **- SSL 종료**  **- 라우팅**  **- WAF 기능**  **- 세션 고정**  **end note**  **}**  **package "API 계층" {**  **[App Service] as appService**  **note right of appService**  **Spring Boot 애플리케이션**  **end note**  **}**  **package "서비스 계층" {**  **[요금제 컨트롤러] as planController**  **[요금제 서비스] as planService**  **[캐시 서비스] as cacheService**  **[요금제 레포지토리] as planRepository**    **note right of planService**  **- 요금제 CRUD 처리**  **- 캐시 정책 적용**  **end note**  **}**  **package "데이터 접근 계층" {**  **[JPA 엔티티] as jpaEntities**  **[JPA 구현체] as jpaImpl**    **note right of jpaEntities**  **@Entity**  **public class UserPlan {**  **@Id Long userId;**  **String planName;**  **String planDetails;**  **LocalDateTime lastUpdated;**  **}**  **end note**  **}**  **package "데이터 계층" {**  **database "Azure Cache\nfor Redis" as redisCache {**  **[Cache Entry] as cacheEntry**  **}**  **database "Azure SQL\nDatabase" as sqlDb {**  **[User\_Plans Table] as userPlansTable**  **}**  **}**  **package "모니터링 계층" {**  **[Azure Monitor] as monitor**  **note right of monitor**  **- 캐시 히트율**  **- DB 성능**  **- 응답시간**  **end note**  **}**  **' 관계 정의**  **mobileApp --> appGateway : HTTPS**  **webApp --> appGateway : HTTPS**  **appGateway --> appService : HTTPS/443**  **appService --> planController : HTTP/8080**  **planController --> planService**  **planService --> cacheService**  **planService --> planRepository**  **planRepository --> jpaEntities**  **jpaEntities --> jpaImpl**  **jpaImpl --> sqlDb : JDBC**  **cacheService --> redisCache : Redis Protocol/6379**  **monitor --> redisCache : 메트릭 수집**  **monitor --> sqlDb : 메트릭 수집**  **monitor --> appGateway : 메트릭 수집**  **note right of cacheService**  **캐시 조회/갱신 로직:**  **1. Redis 캐시 조회**  **2. Cache Miss -> JPA로 DB 조회**  **3. DB 데이터를 캐시에 저장 (TTL: 10분)**  **4. 데이터 반환**  **end note**  **@enduml** |
| --- |

### 2) 물리 아키텍처 예시

**제목: sample\_물리아키텍처**

| **@startuml**  **!theme mono**  **title Resource Group: tiu-dgga-rg - 물리 아키텍처**  **' Azure Resource Group**  **rectangle "Resource Group (tiu-dgga-rg)" {**  **' Virtual Network**  **rectangle "Virtual Network (tiu-dgga-vnet)" {**  **' AKS Cluster**  **rectangle "AKS Cluster" {**  **rectangle "System Node Pool" {**  **[Ingress Controller] as ingress**  **}**    **rectangle "User Node Pool" {**  **rectangle "Command Service Pod" as cmd\_pod {**  **[Command API] as cmd\_api**  **[Spring Boot App] as cmd\_app**  **}**    **rectangle "Query Service Pod" as query\_pod {**  **[Query API] as query\_api**  **[Spring Boot App] as query\_app**  **}**    **rectangle "Database Pod" {**  **database "PostgreSQL\n(Write DB)" as postgres**  **database "MongoDB\n(Read DB)" as mongodb**  **}**  **}**  **}**  **' Subnets**  **rectangle "Subnets" {**  **rectangle "AKS Subnet\n(tiu-dgga-aks-snet)" as aks\_subnet**  **rectangle "Private Subnet\n(tiu-dgga-pri-snet)" as pri\_subnet**  **}**  **}**    **' Azure Event Hub**  **component "Event Hub Namespace" as eventhub {**  **queue "Event Hub" as event\_queue**  **}**    **' Azure Load Balancer**  **component "Azure Load Balancer" as load\_balancer**    **' DNS & Monitoring**  **component "Private DNS Zone" as dns**  **component "Application Insights" as app\_insights**  **}**  **' External Actors**  **actor "Client" as client**  **actor "Developer" as developer**  **' Network Flow Connections**  **client --> load\_balancer**  **load\_balancer --> ingress**  **ingress --> cmd\_pod : "Write\nRequests"**  **ingress --> query\_pod : "Read\nRequests"**  **' Data Flow Connections**  **cmd\_api --> postgres : "Write"**  **cmd\_api --> event\_queue : "Publish Events"**  **event\_queue --> query\_api : "Subscribe Events"**  **query\_api --> mongodb : "Read/Write"**  **' Monitoring Connections**  **cmd\_app ..> app\_insights : "Telemetry"**  **query\_app ..> app\_insights : "Telemetry"**  **' Development Access**  **developer --> aks\_subnet : "kubectl"**  **' Legend**  **legend right**  **| Component | Description |**  **|---|---|**  **| Infrastructure | AKS Cluster with dual node pools |**  **| | Azure Load Balancer for ingress |**  **| | Event Hub for event sourcing |**  **| Application | Command/Query services |**  **| | PostgreSQL for write model |**  **| | MongoDB for read model |**  **| | Application monitoring |**  **endlegend**  **note right of event\_queue**  **Event-driven 아키텍처 구현**  **- 비동기 메시지 처리**  **- 시스템 간 결합도 감소**  **end note**  **note right of app\_insights**  **시스템 모니터링**  **- 성능 메트릭 수집**  **- 로그 분석**  **- 알림 설정**  **end note**  **@enduml** |
| --- |

### 3) API설계서 sample

**제목: sample\_API설계서**

| 서비스명|구독추천|구독추천|구독추천|구독추천|구독추천  마이크로서비스 이름|SubRecommend|SubRecommend|SubRecommend|SubRecommend|SubRecommend  유저스토리 ID|RSS-005|RSS-015|RSS-020|RSS-025|RSS-030  유저스토리 제목|추천 구독 카테고리|지출 카테고리 및 총 지출액|구독 카테고리 표시|구독서비스 목록|구독상세  Controller 이름|RecommendController|RecommendController|RecommendController|RecommendController|RecommendController  API 목적|최고 지출 카테고리 기반 구독 추천|지출 카테고리 및 총액 조회|구독 카테고리 목록 조회|카테고리별 구독서비스 목록 조회|구독 서비스 상세 정보 조회  API Method|GET|GET|GET|GET|GET  API 그룹 Path|/api/recommend|/api/recommend|/api/recommend|/api/recommend|/api/recommend  API Path|/categories|/expense|/categories|/services|/services/{serviceId}  Path <변수유형> <변수명>||||Long serviceId  Query Key|userId|userId||category|  Query <변수유형> <변수명>|String userId|String userId||String category|  Request DTO 이름|||||  Request DTO 배열 여부|||||  Request DTO 구조|||||  Response DTO 이름|RecommendCategoryDTO|ExpenseSummaryDTO|CategoryListDTO|ServiceListDTO|ServiceDetailDTO  Response DTO 배열 여부|No|No|Yes|Yes|No  Response DTO 구조|String categoryName; String imagePath; String baseDate|String topCategory; int totalExpense|String categoryName|String serviceName; String logoPath; String description; int price|String serviceName; String logoPath; String category; String description; int price; int maxSharedUsers |
| --- |

### 4) 시퀀스설계서 예시

#### **가. 외부 시퀀스 설계서**

**제목: sample\_시퀀스설계서(외부)**

| **@startuml**  **!theme mono**  **title 전체 서비스 시퀀스 다이어그램**  **actor Client**  **participant APIGateway**  **participant "회원 서비스" as MemberService**  **participant "구독추천 서비스" as RecommendationService**  **participant "마이구독 서비스" as MySubscriptionService**  **participant "마이그룹 서비스" as MyGroupService**  **participant "이체조회 서비스" as TransactionService**  **' 회원 서비스 호출**  **Client -> APIGateway: 회원 관련 요청**  **APIGateway -> MemberService: /api/auth/signup**  **APIGateway -> MemberService: /api/auth/login**  **APIGateway -> MemberService: /api/auth/verify**  **APIGateway -> MemberService: /api/auth/refresh**  **APIGateway -> MemberService: /api/users/{userId}**  **APIGateway -> MemberService: /api/users**  **APIGateway -> MemberService: /api/auth/logout**  **' 구독추천 서비스 호출**  **Client -> APIGateway: 구독추천 관련 요청**  **APIGateway -> RecommendationService: /api/recommendations/categories**  **APIGateway -> RecommendationService: /api/recommendations/services**  **APIGateway -> RecommendationService: /api/recommendations/expenses**  **APIGateway -> RecommendationService: /api/categories**  **APIGateway -> RecommendationService: /api/services/{serviceId}**  **APIGateway -> RecommendationService: /api/subscriptions**  **' 마이구독 서비스 호출**  **Client -> APIGateway: 마이구독 관련 요청**  **APIGateway -> MySubscriptionService: /api/users/{userId}/subscriptions/summary**  **APIGateway -> MySubscriptionService: /api/users/{userId}/subscriptions/total-fee**  **APIGateway -> MySubscriptionService: /api/users/{userId}/subscriptions/max-savings**  **APIGateway -> MySubscriptionService: /api/users/{userId}/subscriptions**  **APIGateway -> MySubscriptionService: /api/subscriptions/{subscriptionId}**  **APIGateway -> MySubscriptionService: /api/users/{userId}/subscriptions/{subscriptionId}**  **MySubscriptionService -> RecommendationService: 서비스 정보 요청**  **MySubscriptionService -> MyGroupService: 그룹 정보 확인**  **' 마이그룹 서비스 호출**  **Client -> APIGateway: 마이그룹 관련 요청**  **APIGateway -> MyGroupService: /api/users/{userId}/total-subscription-fee**  **APIGateway -> MyGroupService: /api/users/{userId}/max-savings**  **APIGateway -> MyGroupService: /api/users/{userId}/ssups/summary**  **APIGateway -> MyGroupService: /api/users/{userId}/ssups**  **APIGateway -> MyGroupService: /api/ssups/{ssupId}**  **APIGateway -> MyGroupService: /api/ssups**  **APIGateway -> MyGroupService: /api/ssups/join**  **APIGateway -> MyGroupService: /api/users/{userId}/ssups/{ssupId}**  **APIGateway -> MyGroupService: /api/ssups/{ssupId}/transactions**  **MyGroupService -> RecommendationService: 서비스 정보 요청**  **MyGroupService -> MySubscriptionService: 구독 정보 요청**  **' 이체조회 서비스 호출**  **Client -> APIGateway: 이체내역 조회 요청**  **APIGateway -> TransactionService: /api/ssups/{ssupId}/transactions**  **@enduml** |
| --- |

#### 나. 내부 Sequence Diagram

**제목: sample\_시퀀스설계서(내부)**

| **@startuml**  **!theme mono**  **title 마이구독 서비스 상세 시퀀스 다이어그램**  **actor Client**  **participant APIGateway**  **participant "구독 컨트롤러\n(SubscriptionController)" as SubscriptionController**  **participant "구독 서비스\n(SubscriptionService)" as SubscriptionService**  **participant "구독 저장소\n(SubscriptionRepository)" as SubscriptionRepository**  **database "구독 DB" as SubscriptionDB**  **participant "구독추천 서비스" as RecommendationService**  **participant "마이그룹 서비스" as MyGroupService**  **' 구독 요약 조회**  **Client -> APIGateway: GET /api/users/{userId}/subscriptions/summary**  **activate APIGateway**  **APIGateway -> SubscriptionController: 구독 요약 조회 요청**  **activate SubscriptionController**  **SubscriptionController -> SubscriptionService: 사용자 구독 요약 조회**  **SubscriptionService -> SubscriptionRepository: 구독 정보 조회**  **SubscriptionRepository -> SubscriptionDB: 구독 데이터 조회**  **deactivate SubscriptionController**  **deactivate APIGateway**  **' 총 구독료 조회**  **Client -> APIGateway: GET /api/users/{userId}/subscriptions/total-fee**  **activate APIGateway**  **APIGateway -> SubscriptionController: 총 구독료 조회 요청**  **activate SubscriptionController**  **SubscriptionController -> SubscriptionService: 총 구독료 계산**  **SubscriptionService -> SubscriptionRepository: 사용자 구독 정보 조회**  **SubscriptionRepository -> SubscriptionDB: 구독 데이터 조회**  **deactivate SubscriptionController**  **deactivate APIGateway**  **' 최대 절감액 조회**  **Client -> APIGateway: GET /api/users/{userId}/subscriptions/max-savings**  **activate APIGateway**  **APIGateway -> SubscriptionController: 최대 절감액 조회 요청**  **activate SubscriptionController**  **SubscriptionController -> SubscriptionService: 최대 절감액 계산**  **SubscriptionService -> SubscriptionRepository: 사용자 구독 정보 조회**  **SubscriptionRepository -> SubscriptionDB: 구독 데이터 조회**  **deactivate SubscriptionController**  **deactivate APIGateway**  **' 구독 목록 조회**  **Client -> APIGateway: GET /api/users/{userId}/subscriptions**  **activate APIGateway**  **APIGateway -> SubscriptionController: 구독 목록 조회 요청**  **activate SubscriptionController**  **SubscriptionController -> SubscriptionService: 사용자 구독 목록 조회**  **SubscriptionService -> SubscriptionRepository: 구독 정보 조회**  **SubscriptionRepository -> SubscriptionDB: 구독 데이터 조회**  **deactivate SubscriptionController**  **deactivate APIGateway**  **' 구독 상세 조회**  **Client -> APIGateway: GET /api/subscriptions/{subscriptionId}**  **activate APIGateway**  **APIGateway -> SubscriptionController: 구독 상세 조회 요청**  **activate SubscriptionController**  **SubscriptionController -> SubscriptionService: 구독 상세 정보 조회**  **SubscriptionService -> SubscriptionRepository: 구독 정보 조회**  **SubscriptionRepository -> SubscriptionDB: 구독 데이터 조회**  **SubscriptionService -> RecommendationService: 구독 서비스 상세 정보 요청**  **deactivate SubscriptionController**  **deactivate APIGateway**  **' 구독 취소**  **Client -> APIGateway: DELETE /api/users/{userId}/subscriptions/{subscriptionId}**  **activate APIGateway**  **APIGateway -> SubscriptionController: 구독 취소 요청**  **activate SubscriptionController**  **SubscriptionController -> SubscriptionService: 구독 취소 처리**  **SubscriptionService -> SubscriptionRepository: 구독 정보 삭제**  **SubscriptionRepository -> SubscriptionDB: 구독 데이터 삭제**  **SubscriptionService -> MyGroupService: 관련 구독그룹 확인 요청**  **deactivate SubscriptionController**  **deactivate APIGateway**  **@enduml** |
| --- |

### 5) Class 설계 예시

**제목: sample\_클래스설계서**

| **@startuml**  **!theme mono**  **title 마이구독 서비스 - 클래스 다이어그램**  **' Presentation Layer**  **package "Presentation Layer" {**  **class MySubController {**  **-mySubService: IMySubService**  **+getTotalSubFee(userId: String): ResponseEntity<Long>**  **+getMaxSavingAmount(userId: String): ResponseEntity<Long>**  **+getMySubList(userId: String): ResponseEntity<List<SubInfoDTO>>**  **+cancelSub(subId: Long): ResponseEntity<Void>**  **+subscribeSub(subId: Long): ResponseEntity<Void>**  **}**    **class SecurityConfig {**  **-jwtTokenProvider: JwtTokenProvider**  **+securityFilterChain(http: HttpSecurity): SecurityFilterChain**  **+corsConfigurationSource(): CorsConfigurationSource**  **}**    **class JwtAuthenticationFilter {**  **-jwtTokenProvider: JwtTokenProvider**  **+doFilterInternal(request: HttpServletRequest, response: HttpServletResponse, filterChain: FilterChain): void**  **}**  **}**  **' Application Layer**  **package "Application Layer" {**  **interface IMySubService {**  **+getTotalSubFee(userId: String): Long**  **+getMaxSavingAmount(userId: String): Long**  **+getMySubList(userId: String): List<MySubDTO>**  **+cancelSub(subId: Long): void**  **+subscribeSub(subId: Long, userId: String): void**  **}**    **class MySubServiceImpl {**  **-mySubRepository: IMySubRepository**  **+getTotalSubFee(userId: String): Long**  **+getMaxSavingAmount(userId: String): Long**  **+getMySubList(userId: String): List<MySubDTO>**  **+cancelSub(subId: Long): void**  **+subscribeSub(subId: Long, userId: String): void**  **}**    **class JwtTokenProvider {**  **-algorithm: Algorithm**  **+createToken(memberEntity: MemberEntity, authorities: Collection<GrantedAuthority>): JwtTokenDTO**  **+validateToken(token: String): int**  **+getAuthentication(token: String): Authentication**  **}**  **}**  **' Domain Layer**  **package "Domain Layer" {**  **class MySub {**  **-userId: String**  **-subId: Long**  **+getUserId(): String**  **+getSubId(): Long**  **}**    **class MySubDTO {**  **-userId: String**  **-subId: Long**  **+getUserId(): String**  **+getSubId(): Long**  **}**    **class SubInfoDTO {**  **-userId: String**  **-subId: Long**  **+getUserId(): String**  **+getSubId(): Long**  **}**  **}**  **' Infrastructure Layer**  **package "Infrastructure Layer" {**  **interface IMySubRepository {**  **+findByUserId(userId: String): List<MySubEntity>**  **+findById(subId: Long): Optional<MySubEntity>**  **+delete(mySubEntity: MySubEntity): void**  **+save(mySubEntity: MySubEntity): MySubEntity**  **}**    **class MySubEntity {**  **-userId: String**  **-subId: Long**  **+toDomain(): MySub**  **}**    **class LoggingAspect {**  **+logMethodStart(joinPoint: JoinPoint): void**  **+logMethodEnd(joinPoint: JoinPoint, result: Object): void**  **+logMethodException(joinPoint: JoinPoint, exception: Exception): void**  **}**    **class SpringDocConfig {**  **+openAPI(): OpenAPI**  **}**    **class BizException {**  **+BizException(message: String)**  **+BizException(message: String, cause: Throwable)**  **}**    **class InfraException {**  **-code: int**  **+InfraException(message: String)**  **+InfraException(message: String, cause: Throwable)**  **+InfraException(code: int, message: String)**  **+InfraException(code: int, message: String, cause: Throwable)**  **+getCode(): int**  **}**  **}**  **' Relationships**  **MySubController --> IMySubService**  **MySubServiceImpl ..|> IMySubService**  **MySubServiceImpl --> IMySubRepository**  **MySubEntity --> MySub: converts to**  **JwtAuthenticationFilter --> JwtTokenProvider**  **MySubController ..> MySubDTO: uses**  **MySubController ..> SubInfoDTO: uses**  **IMySubRepository ..> MySubEntity: manages**  **@enduml** |
| --- |

### 6) Data 설계 예시

**제목: sample\_데이터설계서**

| @startuml  !theme mono  title 구독 서비스 - 데이터 모델  ' Style configurations  skinparam linetype ortho  hide circle  ' Member domain entities  entity "Members" as members {  \* userId: varchar(50) <<PK>>  --  userName: varchar(100)  bankName: varchar(50)  bankAccount: varchar(50)  characterId: integer  }  entity "Accounts" as accounts {  \* userId: varchar(50) <<PK>>  --  password: varchar(255)  }  entity "Account\_Roles" as account\_roles {  \* userId: varchar(50) <<FK>>  \* role: varchar(20)  --  }  ' Subscription domain entities  entity "Categories" as categories {  \* id: bigint <<PK>>  --  categoryId: varchar(50)  categoryName: varchar(100)  spendingCategory: varchar(50)  }  entity "Subscriptions" as subs {  \* id: bigint <<PK>>  --  name: varchar(100)  description: text  category\_id: bigint <<FK>>  fee: decimal(15,2)  maxShareNum: integer  logo: varchar(255)  }  entity "Spending\_History" as spending {  \* id: bigint <<PK>>  --  userId: varchar(50)  category: varchar(50)  amount: decimal(15,2)  }  ' Subscription group domain entities  entity "My\_Subscriptions" as my\_sub {  \* id: bigint <<PK>>  --  userId: varchar(50)  subId: bigint  }  entity "Subscription\_Groups" as subgroup {  \* groupId: bigint <<PK>>  --  groupName: varchar(100)  subId: bigint  leaderId: varchar(50)  bankName: varchar(50)  bankAccount: varchar(50)  paymentDay: integer  inviteCode: varchar(20)  }  entity "Group\_Members" as subgroup\_member {  \* groupId: bigint <<FK>>  \* memberId: varchar(50)  }  ' Relationships  members ||--|| accounts  accounts ||--o{ account\_roles  categories ||--|{ subs  subs ||--o{ my\_sub  members ||--o{ my\_sub  subs ||--o{ subgroup  members ||--o{ subgroup : leader  subgroup ||--o{ subgroup\_member  members ||--o{ subgroup\_member  @enduml |
| --- |

## 2. Knowledge에 참조 자료 추가

### 1) User story 추가

**제목: Userstory**

**모든 Worksheet를 PDF로 변환하여 업로드**

### 2) 패키지 구조 표준 추가

아래 내용을 복사하여 ‘패키지 구조 표준’이라는 이름으로 추가

<https://docs.google.com/document/d/12EoB9hRnZXHUQ_e12Y3yUtSj5JUbyk7CB9_3c5eVlUk/edit?tab=t.0>

### 3) 인증서비스 샘플 추가

**제목: authentication.java**

| **JWT기반 인증 서비스 샘플입니다.**  **아래 파일의 내용을 올립니다.**  [**https://drive.google.com/file/d/1ieP6oqlEfG7CM-hk21bUkOPzi7wDCWU3/view?usp=drive\_link**](https://drive.google.com/file/d/1ieP6oqlEfG7CM-hk21bUkOPzi7wDCWU3/view?usp=drive_link) |
| --- |

## 3. Instructions에 설계 관련 지시사항 추가

| [최적안 도출 가이드]  'o:'로 시작하면 최적안을 도출하라는 요청임  1) 각자의 생각을 얘기함  2) 의견을 종합하여 동일한 건 한 개만 남기고 비슷한 건 합침  3) 최적안 후보 5개를 선정함  4) 각 최적안 후보 5개에 대해 평가함  5) 최적안 1개를 선정함  6) 1) ~ 5)번 과정을 10번 반복함  7) 최종으로 선정된 최적안을 제시함  ---  [설계요구사항]  'd:'로 시작하면 UML기반의 설계 요청임.  {공통 요구사항}  - '최적안 가이드'대로 토론하여 최적안을 제시  - PlantUML Script 형식으로 답변(단, API 설계서는 파이프로 구분된 CSV형식으로 답변)  - plantuml script 의 테마 'mono'를 사용하고 제목을 표시(아래 예시 참조)  ```  !theme mono  skinparam classAttributeIconSize 0  skinparam classFontSize 12  skinparam classAttributeFontSize 11  title 멤버십 포인트 적립 시스템 - 클래스 다이어그램  ```  {논리 아키텍처 요구사항}  <요청사항>  - 사용자 관점의 컴포넌트 다이어그램 작성  - 유저스토리를 꼼꼼히 분석하여 사용자 Flow를 표시  - 처리순서별 번호 부여 및 간략 설명  - PlantUML Script로 작성  <참고자료>  - Userstory  <예시>  - sample\_논리아키텍처  <응답형식>  - 코드블록  {API설계 요구사항}  <요청사항>  - 서비스명: 유저스토리의 '서비스'컬럼 참조  - 아래 행(row)으로 구성하여 작성  ```  서비스명  마이크로서비스 이름  유저스토리 ID  유저스토리 제목  Controller 이름  API 목적  API Method  API 그룹 Path  API Path  Path <변수유형> <변수명>  Query Key  Query <변수유형> <변수명>  Request DTO 이름  Request DTO 배열 여부  Request DTO 구조  Response DTO 이름  Response DTO 배열 여부  Response DTO 구조  ```  - 요청 데이터 필드가 1개이면 Path 또는 Query String으로 만들고 Request DTO 이름, 배열여부, 구조는 빈 값으로 할 것  - 인증이 필요한 경우 ‘authentication.java’를 참조  <참고자료>  - Userstory  - 논리아키텍처  <예시>  - sample\_API설계서  <응답형식>  - 코드 블록  {외부 시퀀스설계 요구사항}  <요청사항>  - 모든 마이크로서비스와 외부 시스템을 참여자로 추가  - 각 마이크로서비스의 모든 API를 표시할 것. 한글 설명도 보여줄것.  - 마이크로서비스 내부의 처리 흐름은 표시하지 않음  - 요청만 표시하고 응답은 표시하지 말것  - PlantUML Script로 작성  <참고자료>  - Userstory  - 논리아키텍처  - API 설계서  <예시>  - sample\_시퀀스설계서(외부)  <응답형식>  - 코드블록  {내부 시퀀스설계 요구사항}  <요청사항>  - 마이크로서비스의 모든 API를 표시할 것  - 마이크로서비스 내부의 처리 흐름을 표시  - 인증 서비스는 ‘authentication.java’를 참조  - PlantUML Script로 작성  <참고자료>  - Userstory  - 논리아키텍처  - API 설계서  <예시>  - sample\_시퀀스설계서(내부)  <응답형식>  - 코드블록  {클래스설계 요구사항}  <요청사항>  - 멀티프로젝트 구조로 설계  - {아키텍처 패턴}을 적용  - '패키지 구조 표준'을 준용  - Clean아키텍처 적용 시 Port/Adapter라는 용어 대신 Clean 아키텍처에 맞는 용어 사용  - API 설계서의 2번째 컬럼부터 있는 API를 누락하지 말고 모두 반영할 것  - 클래스 간의 관계를 표현: Generalization, Realization, Dependency, Association, Aggregation, Composition  - 시퀀스 설계서의 처리 흐름을 꼼꼼히 반영할 것  - 프라퍼티와 메소드를 모두 기술할 것  - {SERVICE}마다 응답을 분리하고 common은 가장 나중에 제공  - '패키지 구조 표준'의 예시를 참조하여 모든 서비스가 포함된 패키지 구조도를 한꺼번에 제공  - 인증 서비스는 ‘authentication.java’를 참조  - PlantUML Script로 작성  <참고자료>  - Userstory  - 논리아키텍처  - API 설계서  - 시퀀스 설계서  <예시>  - sample\_클래스설계서  <응답형식>  - 코드블록  {데이터설계 요구사항}  <요청사항>  - Class 설계서의 각 서비스별 entity와 일치해야 함  - {SERVICE}마다 데이터 설계를 하고 응답을 각각 분리하여 제공  - 인증 서비스는 ‘authentication.java’를 참조  - PlantUML Script로 작성  <참고자료>  - Userstory  - 논리아키텍처  - API 설계서  - 시퀀스 설계서  - 클래스 설계서  <예시>  - sample\_데이터설계서  <응답형식>  코드블록  {물리 아키텍처 요구사항}  <요청사항>  - Azure Cloud 기반의 배포 아키텍처 작성  - 설계 결과물을 참조하여 최적의 Azure 서비스를 사용  - 처리순서별 번호 부여 및 간략 설명  - PlantUML Script로 작성  <참고자료>  - Userstory  - API 설계서  - 시퀀스 설계서  - 클래스 설계서  - 데이터 설계서  <예시>  - sample\_물리아키텍처  <응답형식>  - 코드블록 |
| --- |

# 논리 아키텍처 설계

## 1. 시작

프롬프트 예시: 사용하는 패턴과 패턴 적용 방안을 제공

| d: 아래 클라우드 디자인 패턴을 적용하여 논리 아키텍처를 설계해 주세요.  - Queue-based Load Leveling패턴 적용: K-rator가 Azure Event Hub에 사용량 정보 발행시 적용  - Cache-Aside 패턴 적용: 사용량정보 업데이트시 Cache key삭제. Cache 조회 실패 시 사용량정보 DB 조회 후 Cache 적재 처리 |
| --- |

※ 참고: local에 plantuml 처리기 실행

docker run -d --name plantuml -p 38080:8080 plantuml/plantuml-server:latest

브라우저에서 [http://localhost:38080으로](about:blank) 접근

## 2. 검토 및 보완

Event storming와 User story를 참고하여 팀원과 AI와 협업하여 논리 아키텍처 완성

## 3. Knowledge에 등록

제목: 논리아키텍처

내용: PlantUML Script 내용 등록

# API 설계

## 1. 시작

| d:API 설계를 해주세요. |
| --- |

## 2. 검토 및 보완

* 응답을 복사하여 엑셀에 붙이고, 파이프로 구분하여 문자열 분리
* AI와 협업하여 완성

## 3. Knowledge에 등록

제목: API설계서

내용: 파이프로 구분된 CSV 내용 등록

# Sequence 설계

## 1. 시작

외부시퀀스 설계

| d: 외부 시퀀스설계를 해 주세요. |
| --- |

내부시퀀스 설계

| d: {서비스}의 내부 시퀀스 설계를 해주세요. |
| --- |

내부 시퀀스 설계는 각 서비스별로 요청합니다.

## 2. 검토 및 보완

* 응답을 복사하여 <https://planttext.com> 또는 local plant uml 처리기에 붙여 검토
* AI와 협업하여 완성

## 3. Knowledge에 등록

제목: 시퀀스 설계서

내용: 외부와 내부 시퀀스 설계 plantuml script를 합쳐서 등록

# Class 설계

## 1. 시작

| d: Class 설계를 해주세요.  - ORG: 회사 또는 조직의 이름  - ROOT: 루트 프로젝트명. 전체 서비스를 통칭하는 이름으로 지정.  - SERVICE: 서비스명을 콤마로 구분하여 지정  - 아키텍처 패턴  - {서비스명}: {Layered 또는 Clean} 중 하나로 지정 |
| --- |

예시)

| d: Class 설계를 해주세요.  - ORG: unicorn  - ROOT: lifesub  - SERVICE: member, mysub, recommend  - 아키텍처 패턴  - member: Layered  - mysub: Clean  - recommend: Layered |
| --- |

## 2. 검토 및 보완

* 응답을 복사하여 <https://planttext.com> 또는 local plant uml 처리기에 붙여 검토
* AI와 협업하여 완성

## 3. Knowledge에 등록

제목: 클래스 설계서

내용: plantuml script을 등록

제목: 패키지 구조도

내용: 제공받은 패키지 구조도 등록

# Data 설계

## 1. 시작

| d: Data 설계를 해 주세요. Class 설계서의 각 서비스별 entity와 일치해야 합니다. |
| --- |

## 2. 검토 및 보완

| o: 클래스 설계서의 각 서비스별 entity와 일치하는지 검토하고 수정된 데이터 설계서를 보여주세요. |
| --- |

* 응답을 복사하여 <https://planttext.com> 또는 local plant uml 처리기에 붙여 검토
* AI와 협업하여 완성

## 3. Knowledge에 등록

제목: 데이터 설계서

내용: plantuml script를 등록

# 물리아키텍처 설계

## 1. 시작

| d: 물리 아키텍처를 설계 해 주세요. 논리 아키텍처와 데이터 설계서와 일치해야 합니다. |
| --- |

## 2. 검토 및 보완

* 응답을 복사하여 <https://planttext.com> 또는 local plant uml 처리기에 붙여 검토
* AI와 협업하여 완성

## 3. Knowledge에 등록

제목: 물리아키텍처

내용: plantuml script를 등록

# 역설계(Reverse Engineering) 가이드

## 1. 사전준비

- 소스를 MergedSource.java로 합쳐서 업로드 또는 GitHub Repository를 바로 연결해도 됨

아래와 같이 mergesrc.py실행 후 project root directory입력

| cd ~/workspace python3 mergesrc.py |
| --- |

- ‘교재’폴더에서 아래 파일을 다운로드하여 업로드

| - sample\_논리아키텍처 - sample\_API설계서 - sample\_시퀀스설계서(외부) - sample\_시퀀스설계서(내부) - sample\_클래스설계서 - sample\_데이터설계서 - sample\_물리아키텍처 |
| --- |

## 2. Instruction에 추가

| [역설계 요구사항]  'rd:'로 시작하면 개발된 코드를 바탕으로 한 Reverse 설계 요청임.  <요구사항>  - '최적화가이드'대로 토론하여 최적안을 제시  - PlantUML Script 형식으로 답변(단, API 설계서는 파이프로 구분된 CSV형식으로 답변)  - plantuml script 의 테마 'mono'를 사용하고 제목을 표시(아래 예시 참조)  <참고자료>  - MergedSource  - sample\_논리아키텍처  - sample\_API설계서  - sample\_시퀀스설계서(외부)  - sample\_시퀀스설계서(내부)  - sample\_클래스설계서  - sample\_데이터설계서  - sample\_물리아키텍처  <응답순서>  1.논리 아키텍처  2.외부 시퀀스설계서  3.서비스별 설계서  - API설계서  - 내부 시퀀스설계서  - 클래스설계서  - 데이터 설계서  4.물리 아키텍처  <응답형식>  - 코드블록 |
| --- |

## 3. 프롬프트

전체 설계서 작성

| rd: 역설계를 해주세요. |
| --- |

특정 설계서만 작성

| rd: 역설계를 해주세요. 서비스별 클래스 설계서를 작성해 주세요. |
| --- |