# **KEA**

Data design

### **Product description**

The product is a platform for deploying, managing, and scaling machine learning models in production. It offers a secure, flexible environment for automating ML tasks like model versioning, routing, and monitoring. With Kubernetes integration and containerization support, it's designed for developers, ML engineers, and enterprises needing scalable, reliable ML infrastructure.

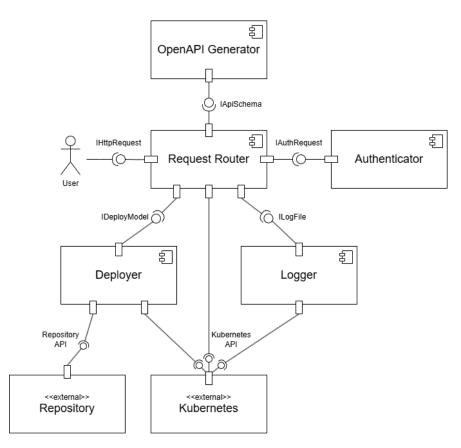
**Team K8C**: Tsurkan Daniel; Dandamaev Gadji; Tsaturyan Konstantin; Smolkin Mikhail

Project repo: <a href="https://github.com/fanglores/Advanced-Software-Design">https://github.com/fanglores/Advanced-Software-Design</a>

This report: <a href="https://github.com/fanglores/Advanced-Software-Design">https://github.com/fanglores/Advanced-Software-Design</a>
/blob/master/Practice%20Tasks/Module2/Task\_12/Task\_12.pdf

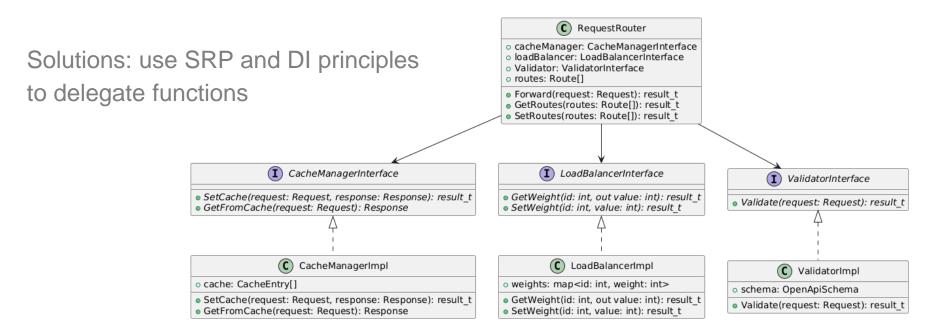
# **System architecture**

BASE Microservices RESTful API



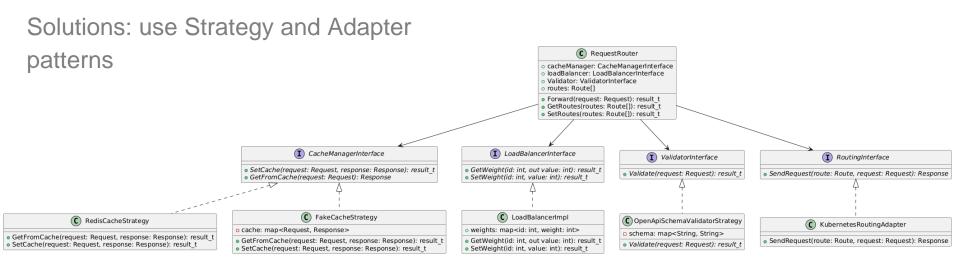
#### Design case for RequestRouter

Problems: many functions for one class



#### Design case for RequestRouter

Problems: strong dependency on Kubernetes, OpenAPI schemas, cache storage



## Design case for Authenticator

Problems: all auth processes are implemented in AuthenticationProvider, new auth methods require changes in a base class

authStrategy: AuthStrategy tokenValidator: TokenValidator tokenCacheManager: TokenCacheManager rulesManager: RulesManagerInterface ValidateToken(token: SSOToken): result t AuthenticateToken(token: String): SSOToken ValidatorInterface AuthInterface SetRule(rule: String): result t ValidateToken(token: SSOToken): result t Authenticate(token: String): result t GetRules(rules: String[]): result t ValidatorImpl C GoogleAuthStrategy o rbacRules: String[] cacheManager: CacheStorageInterface Authenticate(token: String): result t SetRule(rule: String): result t ValidateToken(token: SSOToken): result\_t GetRules(rules: String[]): result\_t (I) CacheStorageInterface Add(token: SSOToken) Get(token: SSOToken): result t (C) FakeCacheStrategy cache: CachedToken[] Add(token: SSOToken) Get(token: SSOToken): result t C CachedToken {singleton} instance: SSOToken CachedToken(token: SSOToken): TokenCache Get(): SSOToken

C AuthenticationProvider

Solution: use SRP and DI principles, Strategy, Singleton patterns for separating different auth-protocols

RulesManagerInterface

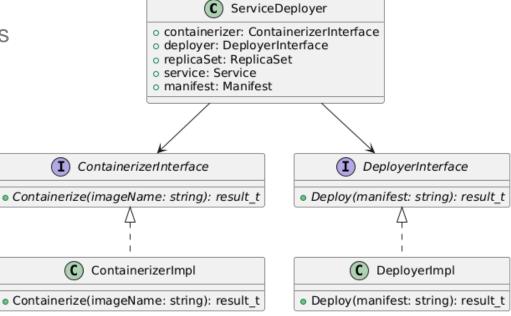
RulesManagerImpl

#### Design case of Service Deployer

Problems: implements many

functions

Solutions: use SRP and DI principles to delegate functions



#### **Design case of Service Deployer**

Problems: new deploy strategies require changes in ServiceDeployer; ServiceDeployer can work with different data, repository or orchestrator

Solutions: use DI principle, Adapter and Strategy patterns

