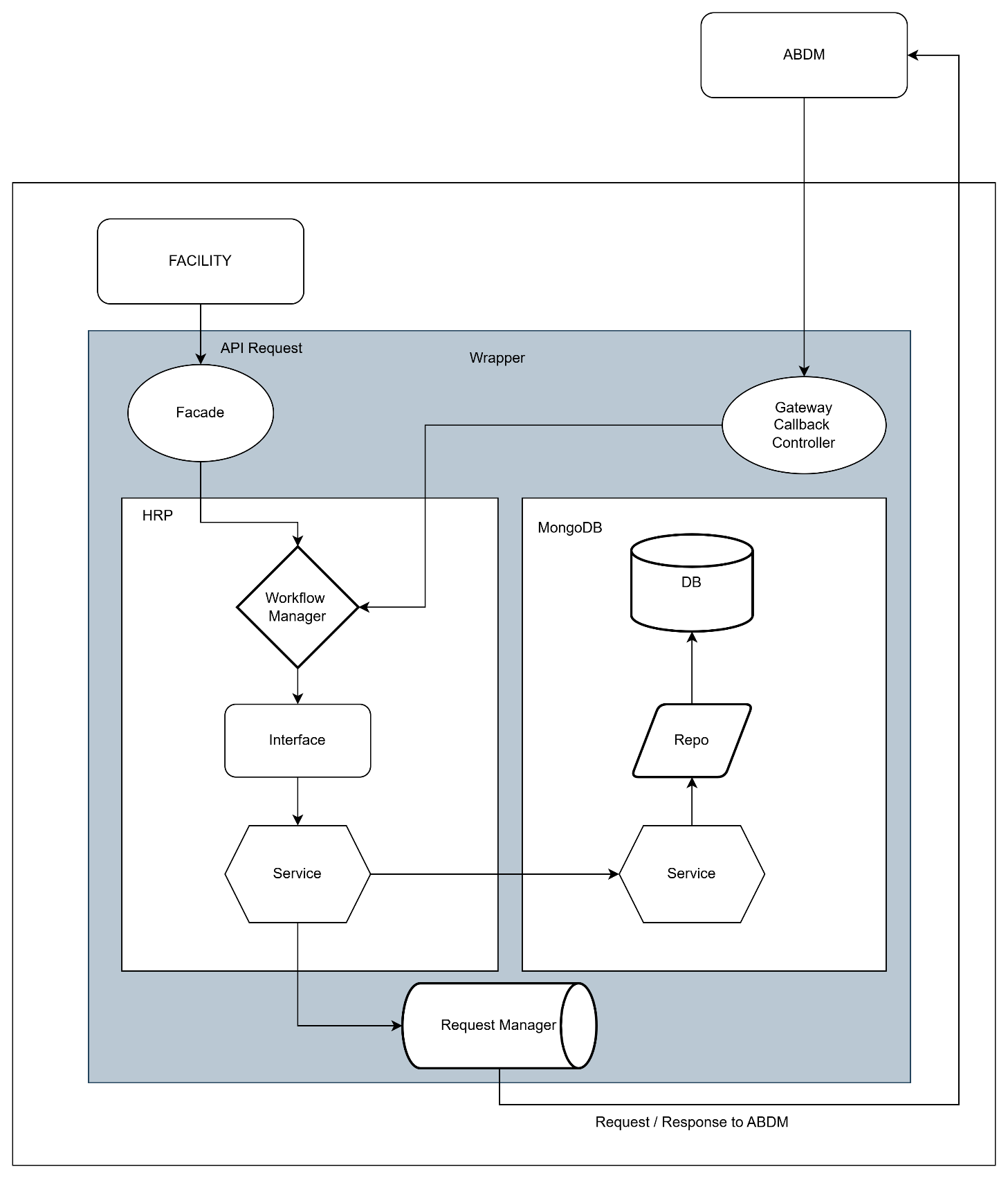
Technical DOC

The wrapper is built upon Facade architecture.

* springBoot: **3.2.0**
* Gradle **8.5**
* Java **17**

**Workflow diagram**



**Let’s start with MongoDB**

MongoDB uses the storage of the server instance, there is no fixed storage allocated to MongoDB,

if the system has 100GB storage it can store 100GB worth data.

There are **7 tables** inside **adbm\_wrapper** MongoDB

* Patient
* request-logs
* link-tokens
* consent-patient
* consent-requests
* consent-careContexts
* consent-cipher-key-mappings

**patient**: Patient table has all the details related to patient, their consent, their careContexts

**request-logs**: All the transaction logs stored with current timestamp in IST format

**link-tokens**: This table consists the linking token with abhaAddress and its expiry

**consent-patient**: This table has the abhaAddress and its related consentId’s

**consent-requests**: This table has the gatewayRequestId along with its clientRequestId

**consent-careContexts**: This table has the list of careContexts related to consent

**consent-cipher-key-mappings**: This table has the keys related to encryption and decryption for both HIU and HIP.

**Docker**

There are two Docker related files

1. Docker file
2. Docker-compose file

**Docker file**: The docker file is written in such a way that, it installs gradle and java and creates a path **/app** and copies all the files present in the working directory and then build and run the JAR file.

**Docker-compose file**: This file is an config file where the services were written in this file.  
 like abdm-wrapper and MongoDB mentioning the port number and the network.

**Example of Linking workflow**

When the facility calls the API: /v3/link-carecontexts

This request will land on the controller present in in/nha/abdm/wrapper/v3/hip/facade/link/**HIPFacadeLinkV3Controller.java**

* ***First Validation Check*:**

With the POJO class annotated with @Valid and @NotNull, spingBoot will check the mandatory attributes, if any field is missing the **GlobalExceptionHandler.java** gets invoked.

* ***Workflow Manager****:*

From controller it gets passed to workflow manager and from here to interface

* ***Interface***: HIPLinkV3Interface.java

In the interface the addCareContexts() gets invoked

* ***Service***: HIPLinkV3Service.java
  + First **patient search**: If found continue for linking or else asking facility for patient details
  + **Checking of careContext**: If care context is present with that particular hiType,

calling /link/context/notify for updation of existing care context.

* + **Check of linkToken**: If token expired or null, requesting ABDM for generation of linkToken and returning back to the facility with 202: Requested successfully
  + After **receiving the linkToken**: updating the DB and initiating the linking of careContext.
  + If Token is not expired, **care context linking is initiated**.
* ***Request Manager:*** RequestV3Manager.java
* All the requests to ABDM are forwarded to this common class for sending POST and GET requests.
* Which consumes webclient from webflux.