Forbis Technical specification.

Proposed Architecture for the NBM Solution

Table of Contents

1. Overview

This document describes the proposed banking system architecture for the National Bank of Moldova tender response. It is aligned with the completed non-functional requirements matrix and explains how the solution satisfies architecture, interoperability, availability, scalability, maintainability, usability, security, auditability, monitoring, backup and support requirements.

1.1 Glossary

Term	Definition
Active Data Guard	Oracle Data Guard option that allows a standby database to be open for read-only workloads while applying redo data from the primary database.
ActiveMQ Artemis	Message broker technology used for asynchronous messaging, queues, topics, reliable delivery, retries, and decoupling between systems.
Apache Camel	Integration framework used to implement routing, transformation, orchestration, protocol mediation, and enterprise integration patterns.
Apache HTTP Server	Web server used to serve frontend static resources, proxy requests, or participate in web-tier traffic handling.
Authentication	Process of verifying the identity of a user, system, or service.
Authorization	Process of determining what functions, data, or operations an authenticated user is allowed to access.
Autonomous Data Guard	Oracle Cloud capability for disaster recovery and standby database protection for Autonomous Database environments.
Autonomous Database	Oracle Cloud managed database service where Oracle automates many operational tasks such as backup, patching, scaling, and availability management.
Back-office	Internal operational banking layer used for administrative, transactional, and data-intensive banking functions.
Business System DMZ	Security zone used for integration, reporting, adapters, external service connectivity, and supporting middleware services.
CI/CD	Continuous Integration / Continuous Delivery- automated process for building, testing, packaging, and deploying application changes.
Client DMZ	Security zone used for customer-facing or external-access components such as remote banking entry points, WAF, and load balancing.
CNF	Compliance or functional/non-functional requirement reference used to link architecture content to tender requirements.
Container	Packaged runtime unit containing an application and its required dependencies, allowing consistent execution across environments.
Control Plane	Kubernetes management layer responsible for cluster state, scheduling, API access, and orchestration control.
Correlation ID	Unique identifier used to trace a transaction or message across multiple services, logs, and systems.
Data Guard	Oracle technology used to maintain standby databases for high availability and disaster recovery.
Data Mart	Subject-specific reporting or analytical data structure derived from operational data.
Database API	Controlled database-level interface, typically implemented through PL/SQL packages or procedures, used by applications to access and modify data safely.
DB Zone	Protected network zone containing Oracle Database, standby database components, database replication, and database backup-related services.

Term	Definition
Dead-Letter Queue	Queue used to store messages that could not be processed successfully after defined retry attempts.
DMZ	Demilitarized Zone- network security zone used to expose controlled services while isolating them from trusted internal systems.
DR	Disaster Recovery- capability to restore or continue critical system operation at a remote site after major failure of the primary site.
ELK	Logging and analytics stack commonly referring to Elasticsearch, Logstash, and Kibana, used for log collection, search, and visualization.
ESB	Enterprise Service Bus- integration architecture layer used to connect systems through routing, transformation, messaging, and service orchestration.
ETL Processes	Data extraction, transformation, validation, reconciliation, and controlled loading processes used for data exchange and migration scenarios.
FBS Data Center	Data center environment hosting the primary or disaster recovery components of the proposed solution.
FBS Database	Central database of the proposed solution containing system kernel data, business data, Oracle Forms repository data, and related subsystem data.
Front-office	User-facing internal banking application layer used by business users to initiate, manage, or process banking activities.
Grafana	Dashboard and visualization tool used to display operational metrics, system health, and performance indicators.
HA	High Availability- architecture approach that minimizes downtime through redundancy, failover, health checks, and load balancing.
Hawtio	Web console used to inspect and manage Java, Camel, and messaging-related runtime components.
HSM	Hardware Security Module- dedicated security device used to protect cryptographic keys and perform secure cryptographic operations.
Identity Provider	System responsible for authenticating users and issuing identity or access tokens to applications.
Ingress Controller	Kubernetes traffic-routing component that receives external or internal HTTP/HTTPS traffic and routes it to the appropriate services inside the cluster.
Internal Zone	Trusted internal network zone containing internal application services, authentication services, back-office middleware, and internal APIs.
ISO/IEC 27001	International standard for information security management systems and security governance.
ISO/IEC 27002	Guidance standard providing implementation recommendations for information security controls.
JasperReports	Reporting technology used to generate PDF and other structured report outputs.
Java REST API Backend	Backend service layer implemented in Java that exposes business functions to React frontend applications and integration channels through REST APIs.
JDBC	Java Database Connectivity- standard Java mechanism used by Java services to connect to Oracle Database.
JBoss EAP	Red Hat-supported enterprise Java application platform used to run Java/Jakarta EE applications.
Jenkins	Automation server used to support build, deployment, and CI/CD processes.
JWT	JSON Web Token- digitally signed token format used to carry identity, session, and authorization-related claims.
Kaoto / Camel Caravan	Tools used to visualize, design, or inspect Apache Camel integration routes and topology.

Term	Definition
Kerberos / SPNEGO	Authentication technologies used to support integrated enterprise authentication and single sign-on scenarios.
Keycloak	Identity and access management component used for authentication, identity federation, token issuing, session management, SSO, and SLO.
Kubernetes	Container orchestration platform used to deploy, scale, manage, and monitor containerized middleware and application services.
Kubernetes Secrets	Kubernetes mechanism used to store and provide sensitive values such as passwords, tokens, certificates, and keys to workloads.
LDAP	Lightweight Directory Access Protocol- protocol used to access directory services such as Microsoft Active Directory.
Load Balancer	Infrastructure component used to distribute traffic across multiple application instances or backend services.
Message Broker	Middleware component used to exchange messages asynchronously between services or systems.
Middle-office	Internal banking application layer supporting control, review, processing, validation, and operational workflows between front-office and back-office functions.
MFA	Multi-Factor Authentication- authentication using more than one factor, such as password, token, certificate, mobile approval, or biometric verification.
mTLS	Mutual TLS; secure communication method where both client and server authenticate each other using certificates.
MS 365 Report Services	Windows-based report generation service used to produce Microsoft Word and Excel report outputs where required.
Namespace	Kubernetes logical separation mechanism used to isolate workloads, configurations, access rights, and resources within a cluster.
NBM	National Bank of Moldova, the target institution for the proposed banking solution.
OAuth 2.0	Open standard authorization framework used for delegated access and token-based security.
OCI	Oracle Cloud Infrastructure- Oracle's cloud platform for compute, networking, database, security, monitoring, and managed cloud services.
OCI Vault	Oracle Cloud service used to manage secrets and cryptographic keys in OCI-based deployments.
OCI Wallet	Oracle wallet mechanism used to store database connection credentials, certificates, and secure connectivity configuration.
OKE	Oracle Kubernetes Engine- managed Kubernetes service in Oracle Cloud Infrastructure.
OLTP	Online Transaction Processing- database workload type optimized for high-volume transactional operations.
OpenAPI / Swagger	Standard format used to document REST APIs and make service interfaces discoverable and testable.
OpenJDK	Open-source Java Development Kit used to run Java-based application components.
Oracle Database 19c Enterprise Edition	Main relational database platform used as the authoritative repository for core banking and subsystem data.
Oracle Forms 14c	Oracle technology used for implementing structured, data-driven, transaction-oriented back-office banking screens and workflows.
Oracle Fusion Middleware	Oracle middleware platform that provides runtime and supporting services for Oracle Forms and related enterprise applications.

Term	Definition
Oracle WebLogic Server	Java application server platform used to run Oracle Forms and related Oracle Fusion Middleware components.
OWASP Top 10	Widely recognized list of major web application security risks used as a baseline for secure application design and validation.
PKI	Public Key Infrastructure- framework for managing digital certificates, keys, and trust relationshFBS.
PL/SQL	Oracle procedural language used to implement database packages, procedures, functions, APIs, validations, and business rules.
Pod	Smallest deployable workload unit in Kubernetes, typically running one or more related containers.
Prometheus	Monitoring and metrics collection system commonly used with Kubernetes and cloud-native applications.
Queue	Messaging structure where messages are stored and processed by one or more consumers, usually for asynchronous workflows.
RAC	Oracle Real Application Clusters - Oracle database clustering technology used to support high availability and workload distribution.
RBAC	Role-Based Access Control- access control model where permissions are assigned through roles or user groups.
React	JavaScript-based frontend technology used to build browser-based user interfaces for front-office, middle-office, and remote banking applications.
Remote Banking	External-facing digital banking channel that provides controlled access to selected banking services for authorized external users or counterparties.
REST API	Application interface style used by frontend and external systems to communicate with backend services over HTTP/HTTPS using structured requests and responses.
RKE2	Rancher Kubernetes Engine 2- enterprise-ready Kubernetes distribution suitable for controlled private, on-premises, or sovereign infrastructure deployments.
RMAN	Oracle Recovery Manager- Oracle-native tool used for database backup, restore, and recovery operations.
RPO	Recovery Point Objective- maximum acceptable amount of data loss measured in time.
RTO	Recovery Time Objective- maximum acceptable time required to restore service after disruption.
SAML 2.0	XML-based identity federation standard used for single sign-on between identity providers and service providers.
SFTP	Secure File Transfer Protocol- secure protocol used for controlled file exchange between systems.
SIEM	Security Information and Event Management- centralized platform for collecting, correlating, and analyzing security-related events.
SLO	Single Logout- identity management principle allowing a user session to be terminated consistently across connected applications.
SMPP	Short Message Peer-to-Peer protocol- protocol commonly used for SMS message exchange with messaging providers.
SMTP	Simple Mail Transfer Protocol- protocol used to send email messages.
SOAP	XML-based web service protocol supported for integrations with systems that require formal service contracts and structured message exchange.
Spring Boot	Java framework used to implement microservices, REST APIs, integration services, and backend application components.

Term	Definition
SSO	Single Sign-On- authentication principle allowing a user to authenticate once and access multiple authorized applications without repeated login.
TAF	Transparent Application Failover- Oracle feature that helps database clients continue operation after certain database connection failures.
TLS	Transport Layer Security- cryptographic protocol used to secure communication between systems.
Topic	Messaging structure used to publish messages to multiple subscribers.
VDI	Virtual Desktop Infrastructure- workplace environment where users access applications through centrally managed virtual desktops.
VMware Tanzu	Enterprise Kubernetes platform option that may be considered if selected or approved by NBM as part of its infrastructure strategy.
WAF	Web Application Firewall- security component used to inspect and protect web traffic against common application-level threats.
WildFly	Open-source Java/Jakarta EE application server and upstream technology related to JBoss EAP.
Worker Node	Kubernetes cluster node where application workloads, pods, and services are executed.
Workload Zone	Logical Kubernetes deployment boundary used to group related workloads, usually implemented through namespaces, node pools, network policies, RBAC, secrets, and resource quotas.
WSDL / XSD	Standards used to describe SOAP web services and XML message structures.

2. Executive Architecture Summary (CNF.4, CNF.10, CNF.14)

The proposed NBM solution is a modular banking platform using a modern layered architecture, secure component communication and standardized integration mechanisms. It separates presentation, business logic and data management responsibilities while allowing each layer to be scaled, secured, maintained and monitored according to its role.

- Back-office administrative performing banking functions are provided access through Oracle Forms under controlled internal access procedures.
- Front-office, middle-office and reporting users' functions are delivered through React-based browser interfaces connected to Java/Web API services.
- Remote banking uses the same modern web technology stack, with customer authentication adapted to the Moldovan public authentication ecosystem.
- Core business rules, validation and transaction consistency are implemented centrally in Oracle database APIs and procedures.
- Integration with NBM systems, payment infrastructures and third parties is performed through a dedicated microservice integration platform based on Spring Boot, Apache Camel and modern Kubernetes-based platform.
- High availability and disaster recovery are addressed through redundant application nodes, load balancing, Oracle RAC/local standby and Oracle Active Data Guard for remote standby database replication.

Supplier solution covers the main technical evaluation areas:

- Modularity – three-layer architecture with separated presentation, business logic and data access; modular internal systems, integration platform and remote banking channel.

- Security - Network zoning, TLS, role-based access, Keycloak /MFA, Oracle authentication, secure APIs, audit trails, least-privilege operation and support for SIEM integration.
- Availability and Disaster Recovery - Primary and disaster recovery geosites with active-passive operating model, redundant application nodes, Oracle RAC/local standby and Active Data Guard remote standby.
- Scalability - Horizontal scaling for stateless services on Kubernetes-based platform; scalable integration microservices; database scaling through Oracle sizing, RAC and data management techniques.
- Operations - Monitoring, logging, archiving and structured incident/support processes.

2.1 General Architectural Principles (CNF.110, CNF.152)

- The system consists of three main components: the internal system, which includes the database layer, application layer and user interface layer; the integration platform; and the remote banking application.
- The system provides the capability to extend each of its components separately.
- The system supports universal integration mechanisms enabling data to be sent to and received from external web services, as well as the development of custom integrations in line with microservices architecture principles. When providing web services, both synchronous and asynchronous data processing is supported.
- The system has data exchange channels with both internal and external systems. The system supports integrations with external systems that provide web services and also provides web services to external systems. Services can be either one-way or two-way.
- The system integration platform ensures both two-way and one-way data exchange with other systems.
- All components of the system have a unified data storage centre — the database. The database provides data to, and receives data from, each system component through a separate dedicated API. Data uniqueness and mandatory data requirements are ensured by database-level mechanisms. Any changes to data, system time or data format in the database are immediately visible to all system components after the change is made. Data exchange errors between individual system components are recorded on the database side in the common system log.
- The system ensures business data uniqueness at the database layer, preventing duplicate records.
- The system components support load balancing.
- The operation and performance of the system are not affected by processes running in parallel sessions.
- The system supports English and Romanian language characters.
- The system provides the capability to configure the required date and number formats.
- The system user interfaces use tooltFBS to help users more easily use the system functions.
- The system supports data import into the system and export from the system in XML format, enabling operation based on a request/response principle. The system also allows data import/export in various other data formats, including CSV, Excel, JSON and TXT, both manually and through automated data exchange, enabling files to be retrieved from and placed on servers using SFTP and FTP protocols.
- Separate infrastructure components of the system communicate using HTTP or HTTPS protocols. The system also uses the functionality provided by the Java application programming interface for database connectivity, namely JDBC.
- The system is implemented using an Oracle database supporting the UTF-8 Unicode standard, in which the accounting data of all business subsystems, as well as data of separate system components, is collected and stored.
- Each subsystem has its own user interface. System users are divided into user groups. Access to subsystems is granted based on user group permissions for specific subsystem functions. Such permissions define whether a user is allowed or prohibited from performing a specific action, or from

viewing and modifying specific subsystem information. These permissions are granted or revoked only by user(s) holding the system administrator role. The system also enforces strict access control to system data based on user type, depending on whether the user is a system user or a system administrator.

- The system supports the execution of automatic tasks, allowing the required actions to be performed on system data and the required execution frequency to be configured. This enables the asynchronous execution of required business function processes.
- The system logs errors that occur while working with a specific business subsystem, as well as failures occurring during data exchange between internal system components. Both automatic task errors and errors occurring during user interaction with the system are recorded.
- The system records information entered, modified or deleted by users, indicating the user and the system time when the specific action was performed.
- The system supports HTTPS, HTTP, SOAP and REST protocols.
- The system supports the capability to work with message brokers such as ActiveMQ, RabbitMQ and others.
- The system supports integrations with SMTP and SMPP servers.
- The system supports correct execution of automatic tasks during time change periods.
- The main business subsystems of the system provide parameter configuration capabilities, allowing subsystem operation to be adjusted according to the defined settings. Such parameters are changed only by users holding the system administrator role.
- The system provides the capability to create new reports without software changes, using only system configuration: by defining the query text and setting the report document format, including MS Word, MS Excel, XML, HTML and TXT.
- The security of individual system modules complies with OWASP Top 10 requirements.

The architecture is designed to minimize the complexity of application-level changes by separating the system into clearly defined components, subsystems, APIs, integration services, configuration parameters, automatic tasks, and reporting mechanisms.

Changes can therefore be implemented in the relevant component without unnecessary impact on unrelated parts of the system. For each change, the affected components are identified based on the changed business function, subsystem, API, database object, integration interface, report, user role, configuration parameter, or scheduled task. This allows targeted testing and validation of only the impacted areas, while still performing regression testing where required by the change risk.

The solution supports several customization categories, including integration changes, user interface changes, report creation, configuration of subsystem parameters, automatic task configuration, data import/export changes, workflow related changes, bank products configuration and data schema or property extensions, where supported by the relevant module. The solution follows TOGAF principles such as modularization, separation of concerns and architecture governance. Architecture development aligns with ADM phases, ensuring traceability from business requirements to implementation.

3. Target Solution Topology (CNF.2, CNF.5, CNF.10, CNF.52, CNF.53, CNF.55, CNF.57, CNF.136, CNF.137, CNF.139, CNF.140, CNF.141, CNF.147)

3.1 High-Level Target Topology

The proposed architecture is organized into logical and network zones that separate external access, business system services, internal processing and database persistence. This separation supports secure access control, clear operational responsibilities and controlled traffic between components.

Client DMZ (external access): remote banking interface entry points, WAF, load balancing and public external users access controls.

Business System DMZ: reporting, ESB, external service adapters and supporting services required to communicate with external parties.

Internal zone: internal system services, Web/API services, authentication services and back-office middleware.

DB zone: Oracle database services, local standby and replication towards the disaster recovery site.

Disaster recovery data center: remote standby database and recovery instances for the critical system components.

The following conceptual topology represents the vendor-neutral supplier target architecture.

The proposed architecture is designed to support deployment on NBM-approved middleware and container platforms.

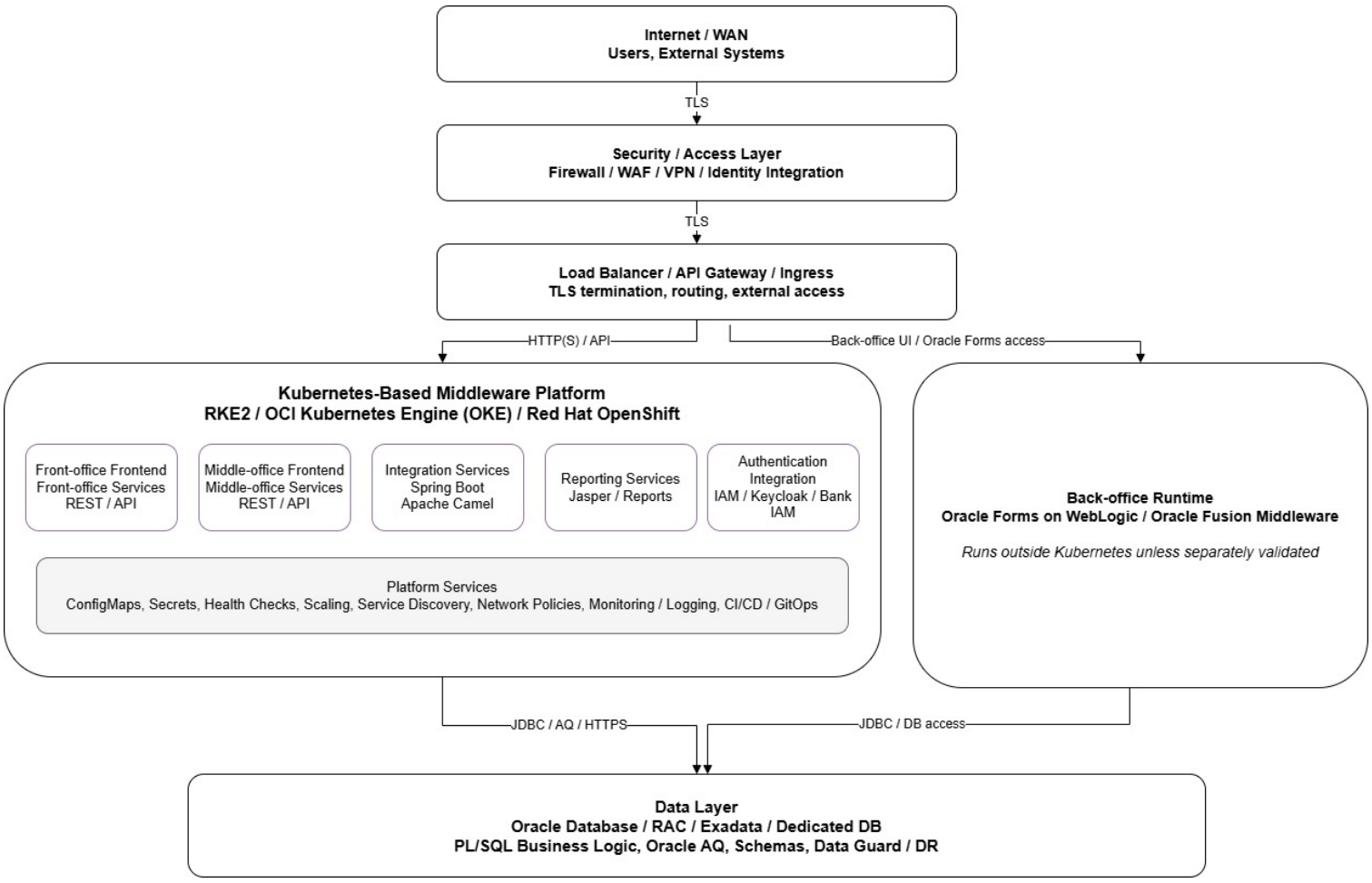


Figure 1. 3.1-1 “Common solution topology”

3.2 Middleware Deployment Flexibility

The proposed middleware architecture is based on a Kubernetes-ready deployment model. Middleware applications are containerized and can be prepared to run on Kubernetes infrastructure using standard deployment principles such as container images, externalized configuration, service discovery, ingress/load balancing, health checks, scaling, monitoring, and controlled release management.

The middleware layer is designed to remain infrastructure flexible. Depending on NBM's preferred infrastructure strategy and the final deployment architecture agreed with NBM, the middleware layer can be adapted for deployment on enterprise Kubernetes platforms such as:

Our first-choice suggested infrastructure configuration, included in the Tender price:

- **Rancher RKE2**, for a secure, lightweight, and fully controlled Kubernetes environment on private or on-premises infrastructure.

Other **optional configuration versions**, presented according to the Tender requirements for your consideration. **Respective licensed are not included** in the Tender price.

- **Red Hat OpenShift**, for an on-premise enterprise Kubernetes environment;
- **Oracle Cloud Infrastructure Container Engine for Kubernetes (OKE)**, for an Oracle-oriented cloud;

This approach allows NBM to select the infrastructure platform that best fits its internal IT strategy, security requirements, regulatory constraints, operational model, and existing vendor ecosystem, including the possibility to use another Kubernetes platform, such as **VMware Tanzu**, if preferred by NBM. The proposed middleware architecture is designed for standard enterprise x86 infrastructure and can be deployed in virtualized and/or Kubernetes-based environments, subject to detailed design, configuration, integration, and deployment decisions agreed with NBM.

3.2.1 Rancher RKE2-based solution – go-to option, included in the Tender price

RKE2 is a strong choice if NBM wants to retain full control over the Kubernetes environment, infrastructure, security configuration, and data location. **This is suggested infrastructure configuration, complimentary licenses are included in the price.**

RKE2 is well suited for institutions that prefer a private-cloud or on-premises deployment model, where infrastructure is managed under the bank's own operational and security governance. It provides a clean, enterprise-ready Kubernetes distribution that can be deployed in controlled environments without forcing dependency on a specific public cloud provider.

For NBM, this option is attractive when data sovereignty, infrastructure independence, and internal operational control are top priorities. It allows the bank to keep the platform close to its existing infrastructure standards while still benefiting from a modern Kubernetes architecture.

Advantages for NBM:

- Strong control over infrastructure, networking, security policies, and deployment location.
- Suitable for on-premises, private cloud, or sovereign infrastructure models.
- Lower dependency on a single public cloud provider.
- Good fit for regulated environments where the bank wants direct control of the platform lifecycle.
- RKE2 has official hardening guidance aligned with CIS (Center for Internet Security) Kubernetes benchmark controls, which is relevant for security-sensitive environments.

Conditions to consider:

- NBM, or its infrastructure partner, must operate and maintain the Kubernetes platform.
- Internal teams need sufficient Kubernetes administration capability.
- High availability, backup, monitoring, patching, and disaster recovery must be designed and operated by the implementation team.
- This option gives maximum control but also requires stronger internal operational responsibility.

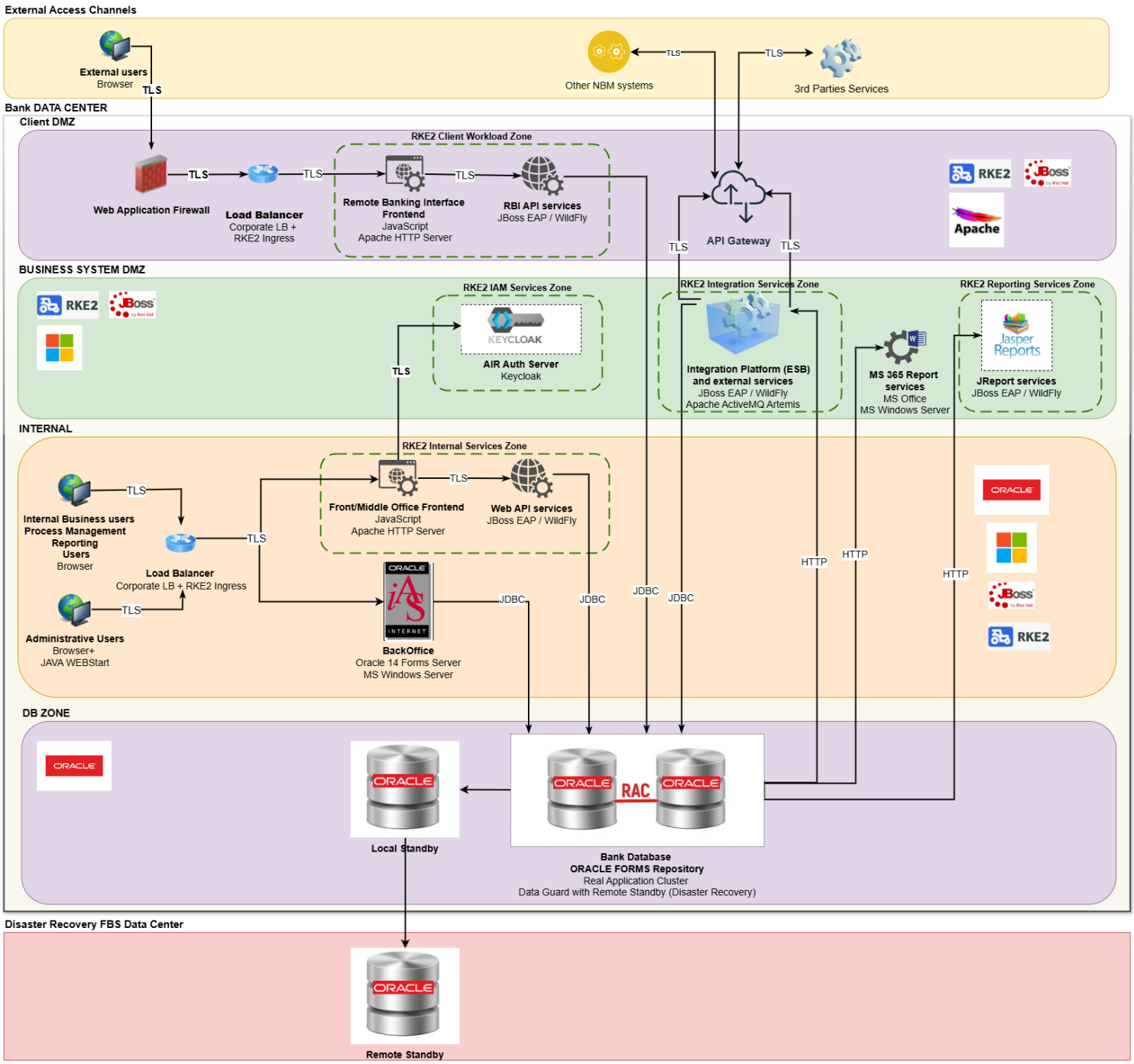


Figure 2. 3.2.1-1 “RKE2-based solution”

3.2.2 Red Hat-based solution - optional, not included in the price

Red Hat OpenShift is a strong choice if NBM wants a complete enterprise Kubernetes platform with mature security, governance, developer tooling, and hybrid-cloud capabilities.

OpenShift is more than a Kubernetes distribution. It provides an integrated enterprise platform for building, deploying, securing, and operating applications across hybrid infrastructure. Red Hat positions OpenShift as a platform for building, modernizing, and deploying applications at scale on the customer's choice of hybrid cloud infrastructure.

For NBM, OpenShift is attractive when the bank wants a standardized enterprise platform with strong vendor support, integrated lifecycle management, security controls, and a mature ecosystem. It is particularly suitable for organizations that want to industrialize application delivery and establish a long-term platform for multiple digital services.

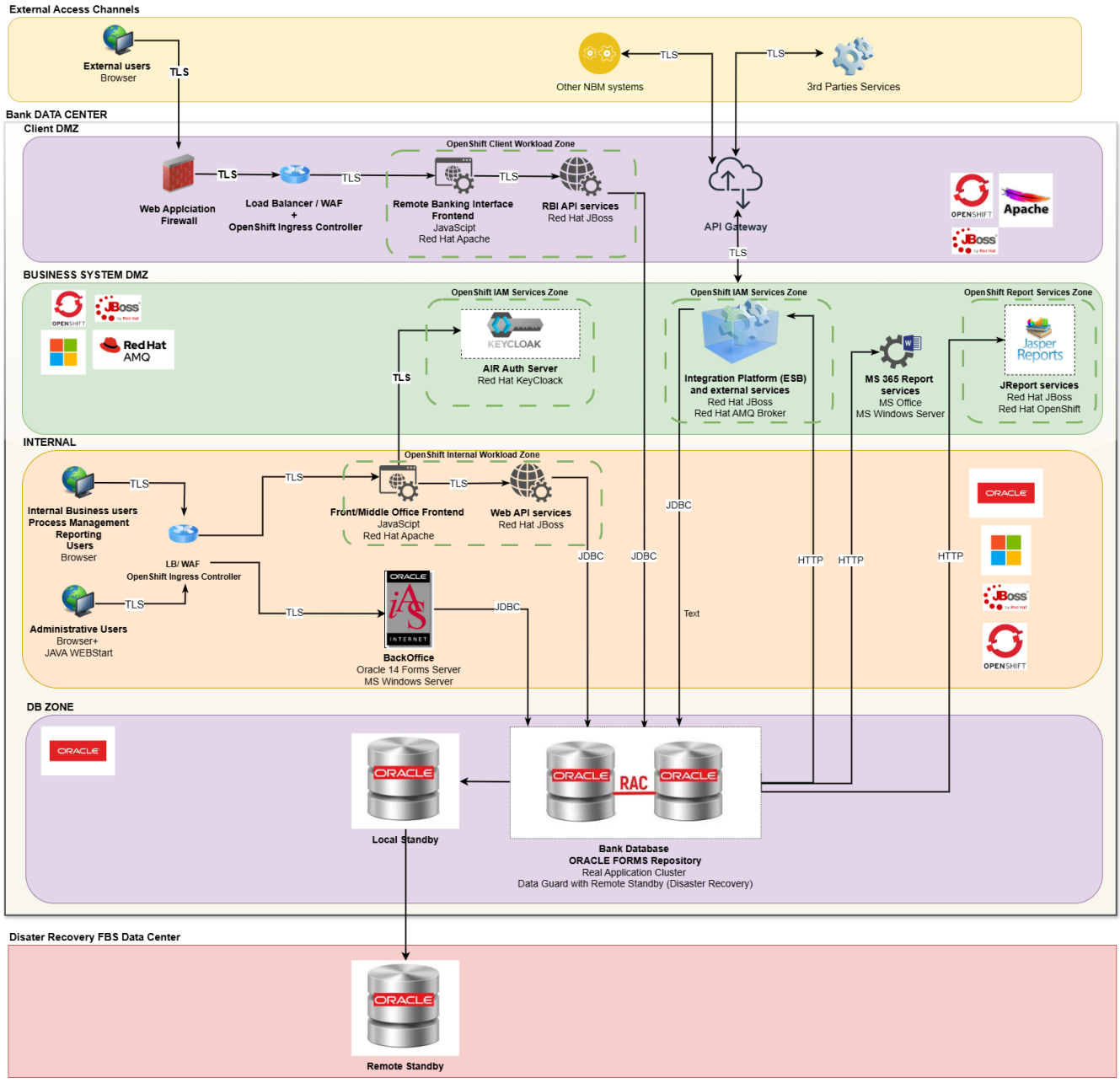


Figure 3. 3.2.2-1 “Red Hat-based solution

Advantages for NBM:

- Enterprise-grade Kubernetes platform with strong vendor support.
- Suitable for hybrid cloud, private cloud, and multi-cloud strategies.
- Integrated platform capabilities for operations, security, application delivery, and governance.
- Strong fit for regulated institutions that need standardization, auditability, and platform maturity.
- OpenShift includes security and compliance capabilities, including documentation around cluster security, certificates, encryption, and compliance operations.

Conditions to consider:

- Requires Red Hat subscription/licensing model.
- Platform is more feature-rich, but also more complex than a lightweight Kubernetes distribution.
- Requires OpenShift-specific skills for administration and operations.
- Best value is achieved when NBM wants a long-term enterprise application platform, not only a basic Kubernetes cluster.

3.2.3 Oracle Cloud Infrastructure (OCI)-based solution - optional, not included in the price

OCI Kubernetes Engine, also known as OKE, is a strong choice if NBM wants a managed Kubernetes service with reduced operational burden and tight integration with Oracle Cloud Infrastructure.

This option is especially attractive if NBM already uses Oracle technologies or plans to use Oracle Cloud services as part of its digital infrastructure strategy. With OKE, the Kubernetes control plane is managed by Oracle, which can reduce the complexity of platform operations and allow NBM teams to focus more on applications, integrations, and business services.

OKE also provides built-in cloud security capabilities such as encryption at rest, private Kubernetes clusters, network security groups, pod-level isolation, RBAC integration with OCI IAM, image scanning/signing, workload identity, and OCI audit services.

Advantages for NBM:

- Managed Kubernetes service reduces operational overhead.
- Strong fit if NBM uses or plans to use Oracle Cloud Infrastructure.
- Built-in integration with OCI networking, IAM, audit, monitoring, storage, and security services.
- Faster provisioning and scaling compared with a fully self-managed Kubernetes environment.
- Good option when NBM wants cloud-native capabilities without managing the full Kubernetes control plane itself.

Conditions to consider:

- Requires strategic acceptance of OCI as the target cloud platform.
- Data residency, regulatory requirements, and cloud governance must be validated against NBM policies.
- Network connectivity between NBM systems and OCI must be carefully designed.

- Cost management, cloud security policies, and service-level responsibilities must be clearly defined.

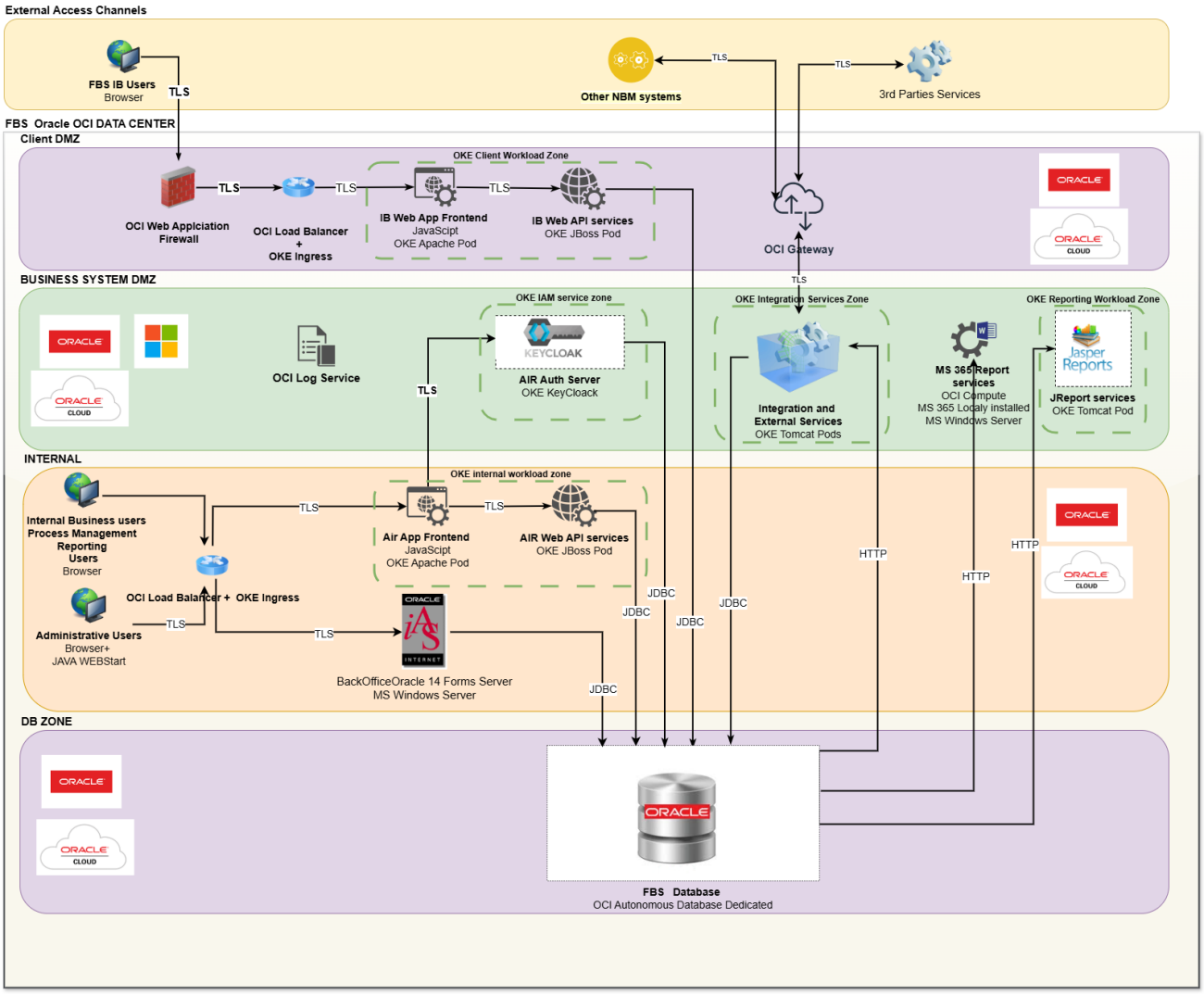


Figure 4. 3.2.3-1“Oracle Cloud-based solution”

3.3 Oracle Forms and Back-Office Flexibility

The current Forbis back-office solution is based on **Oracle Forms**, which can continue to be used in a supported Oracle Forms / Oracle WebLogic / Oracle Fusion Middleware environment.

At the same time, supplier **maintains architectural flexibility**. If required by the NBM’s modernization strategy, supplier can propose an alternative back-office approach based on the same technology direction used for the middle-office and front-office layers. This would allow the NBM to gradually reduce dependency on Oracle Forms and move toward a more unified application architecture across back-office, middle-office, and front-office components.

This ensures that the NBM may choose either:

1. Continue with Oracle Forms for back-office operations, or
2. Modernize selected back-office functions using the same technology stack as the middle-office and front-office layers.

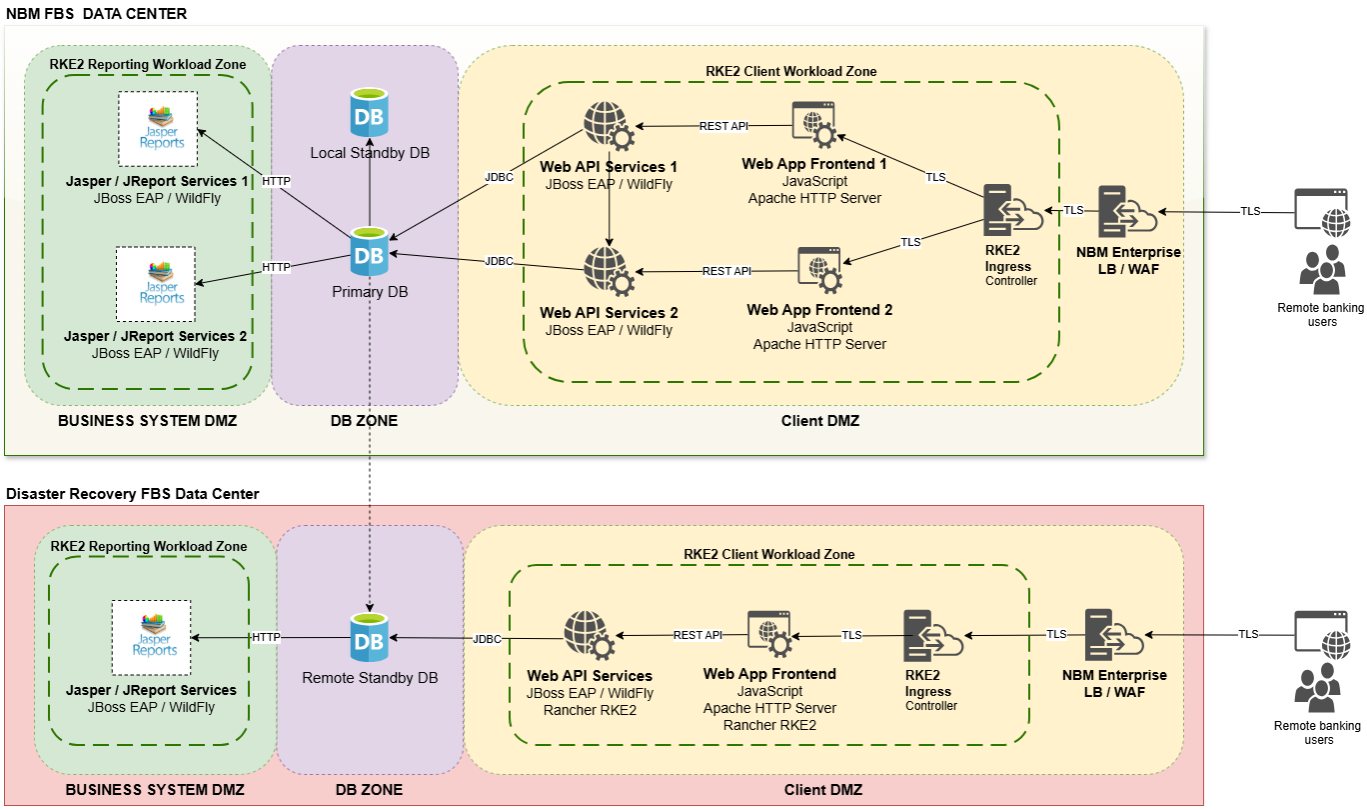
The final approach can be agreed during the detailed solution design phase, taking into account the NBM’s functional scope, modernization priorities, operational readiness, and implementation timeline.

3.4 Remote banking architecture (CNF.10, CNF.152)

The proposed **Remote Banking Application** is a modern external-facing channel designed to provide secure access to selected banking services for authorized external users and counterparties. The solution is implemented using a modern web architecture, with a **React-based user interface** and a Java REST API backend. The Remote Banking backend exposes controlled business services through REST APIs and communicates with the database layer using dedicated backend functions. Remote Banking database objects and backend functions are separated in a dedicated **Oracle schema**, supporting logical separation, maintainability, access control, and clear operational governance. The application includes user-session management mechanisms, including session creation, validation, expiration, and termination controls. Access to Remote Banking functions is controlled according to the user's identity, role, permissions, and allowed business operations. The Remote Banking Application is designed to integrate with the authentication provider required by NBM. Depending on the final security architecture agreed with NBM, authentication may be integrated with the NBM identity provider or another approved authentication mechanism. The application architecture supports secure authentication, authorization, session handling, audit logging, and controlled access to exposed business functions. The Remote Banking Application communicates with the Internal Banking Application and Integration Platform through controlled APIs and agreed security mechanisms. This ensures that external users do not access internal back-office components directly, while all business operations remain validated, logged, and processed through the approved system architecture.

RKE2-based solution:

Figure 5. 3.4-1 "RKE2-based solution"



Red Hat-based solution:

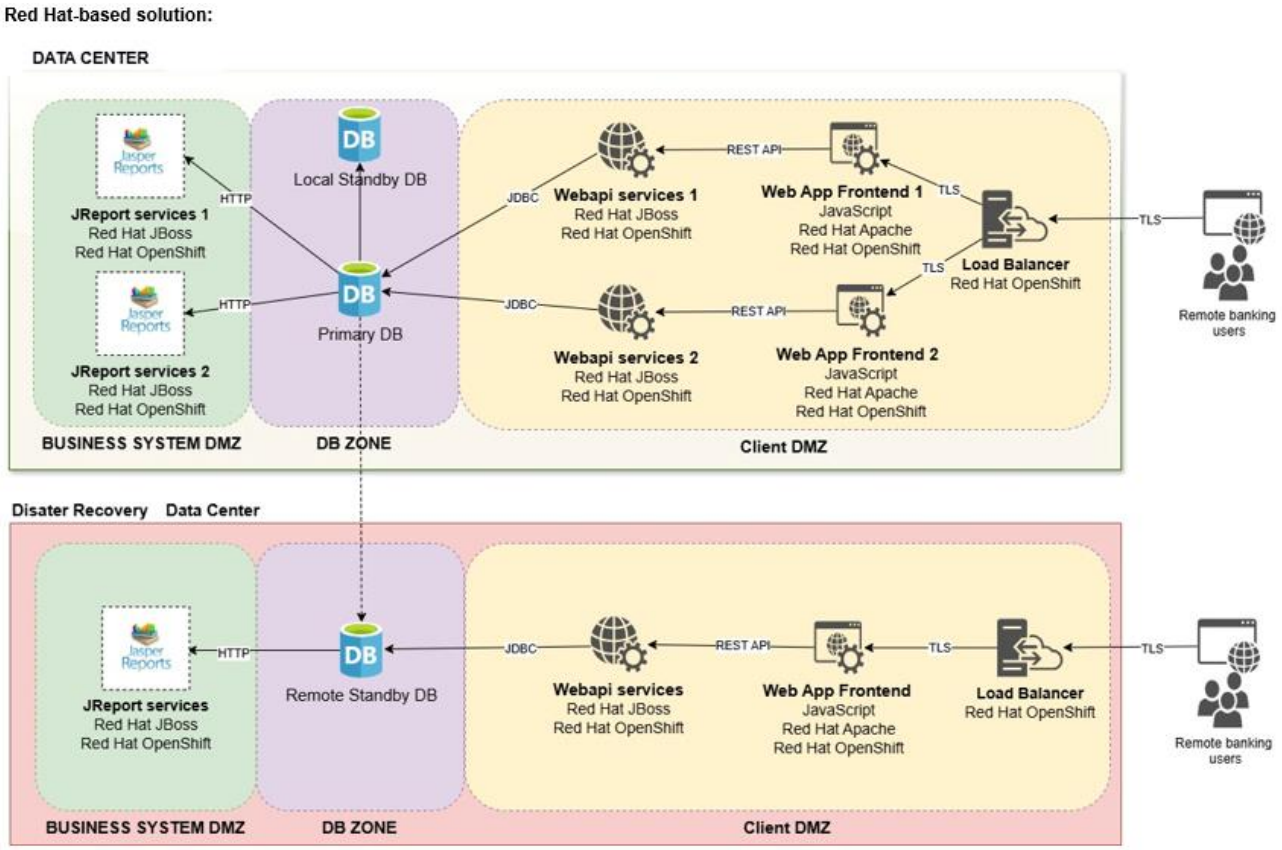


Figure 6. 3.4-2 “Red Hat-based solution”

Oracle Cloud(OCI) solution:

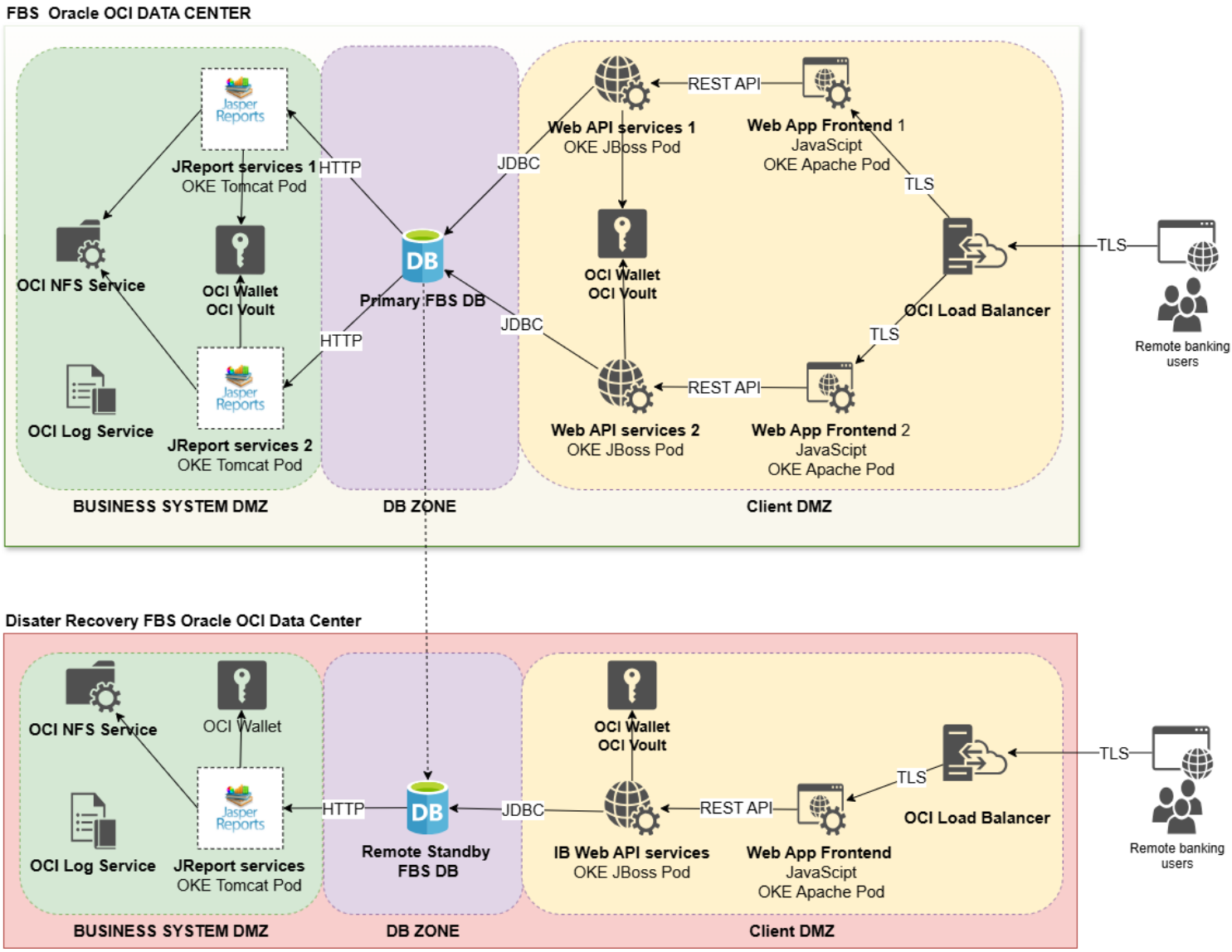


Figure 7. 3.4-3 “Oracle Cloud-based solution”

3.5 Back-office architecture

The Back-office solution is implemented using Oracle Forms 14c, leveraging its mature enterprise capabilities for secure, data-driven, and transaction-oriented banking operations. The proposed approach enables implementation of complex Back-office functions through structured user interfaces, tight Oracle Database integration, and controlled execution of operational workflows, while preserving stability, reliability, and maintainability of critical banking processes.

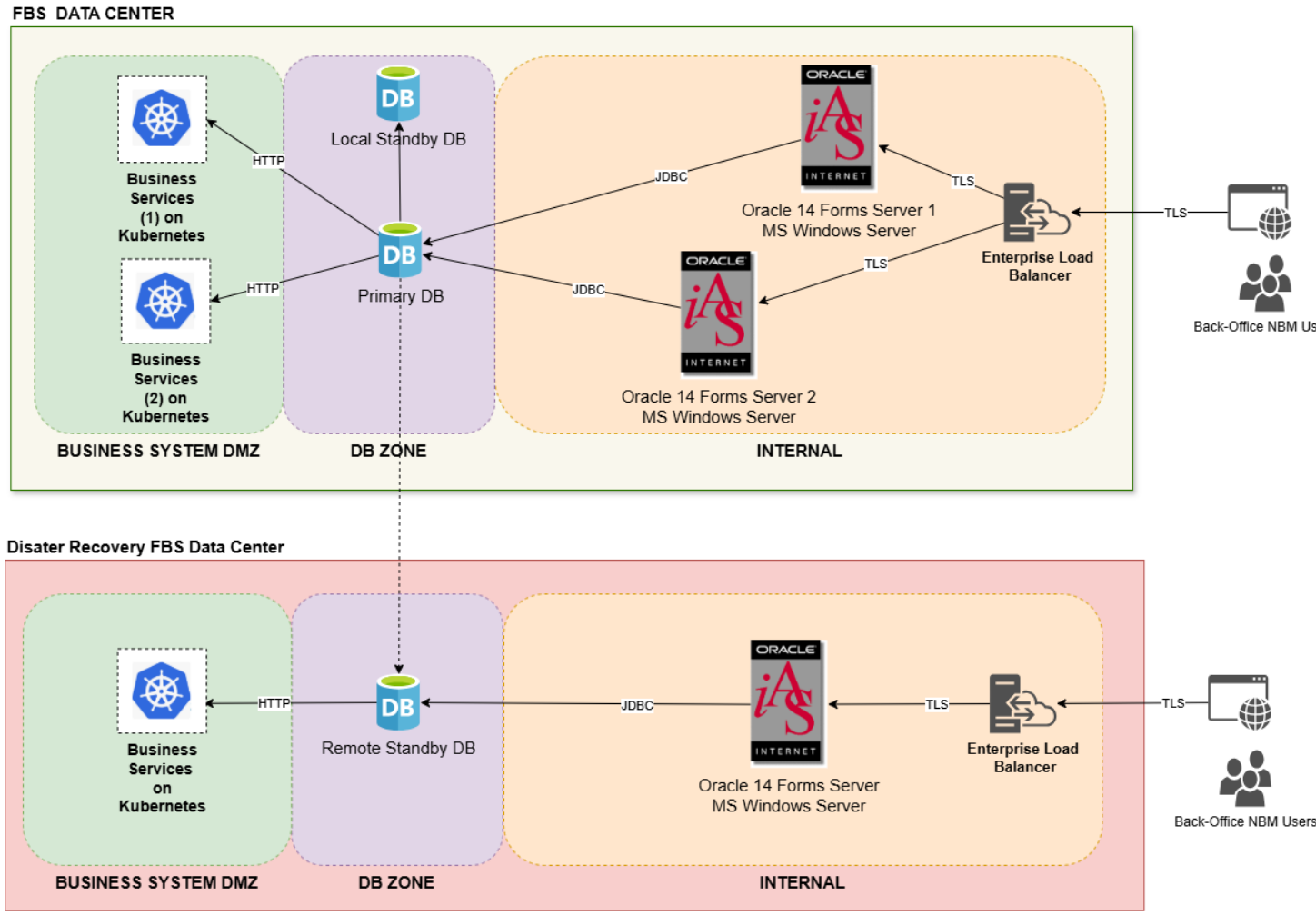


Figure 8. 3.5-1 “Back-office solution”

3.6 Front-office and middle-office architecture

The proposed Front-office and Middle-office solution is designed as a modern, web-based architecture that separates user interface, service logic, identity management, and business authorization responsibilities. The user-facing layer is implemented using a **React-based frontend**, providing a responsive browser interface for Front-office and Middle-office users. The frontend communicates with the backend through a controlled **REST API layer**. All REST service functionality is implemented in dedicated backend components, with business functions and data access organized through separate database schemas. This approach supports clear separation of responsibilities, maintainability, scalability, and controlled access to business data. Authentication and session management are centralized through **Keycloak**, which acts as the common Identity Provider for two separate applications: the **Front-office application** and the **Middle-office application**. The solution supports **Single Sign-On (SSO)** and **Single Logout (SLO)** principles, enabling users to authenticate once and securely access authorized functions across both applications. In this architecture, **Keycloak is responsible for authentication**, identity federation, token issuing, and session handling. Business-level authorization is managed separately within the backend and database business logic, where user roles, permissions, and functional access rules are validated according to the bank’s operational and security requirements. This design provides NBM with a secure and flexible Front-office and Middle-office platform that can be deployed consistently across supported Kubernetes environments such as **RKE2**, **OCI Kubernetes Engine**, or **Red Hat OpenShift**, while preserving the same application architecture, REST integration model, and centralized identity management approach. The following schema presents one representative application topology, as the Front-office and Middle-office applications are separate applications but use the same deployment

pattern, REST API interaction model, Keycloak-based authentication, and database-driven authorization approach.

RKE2-based solution:

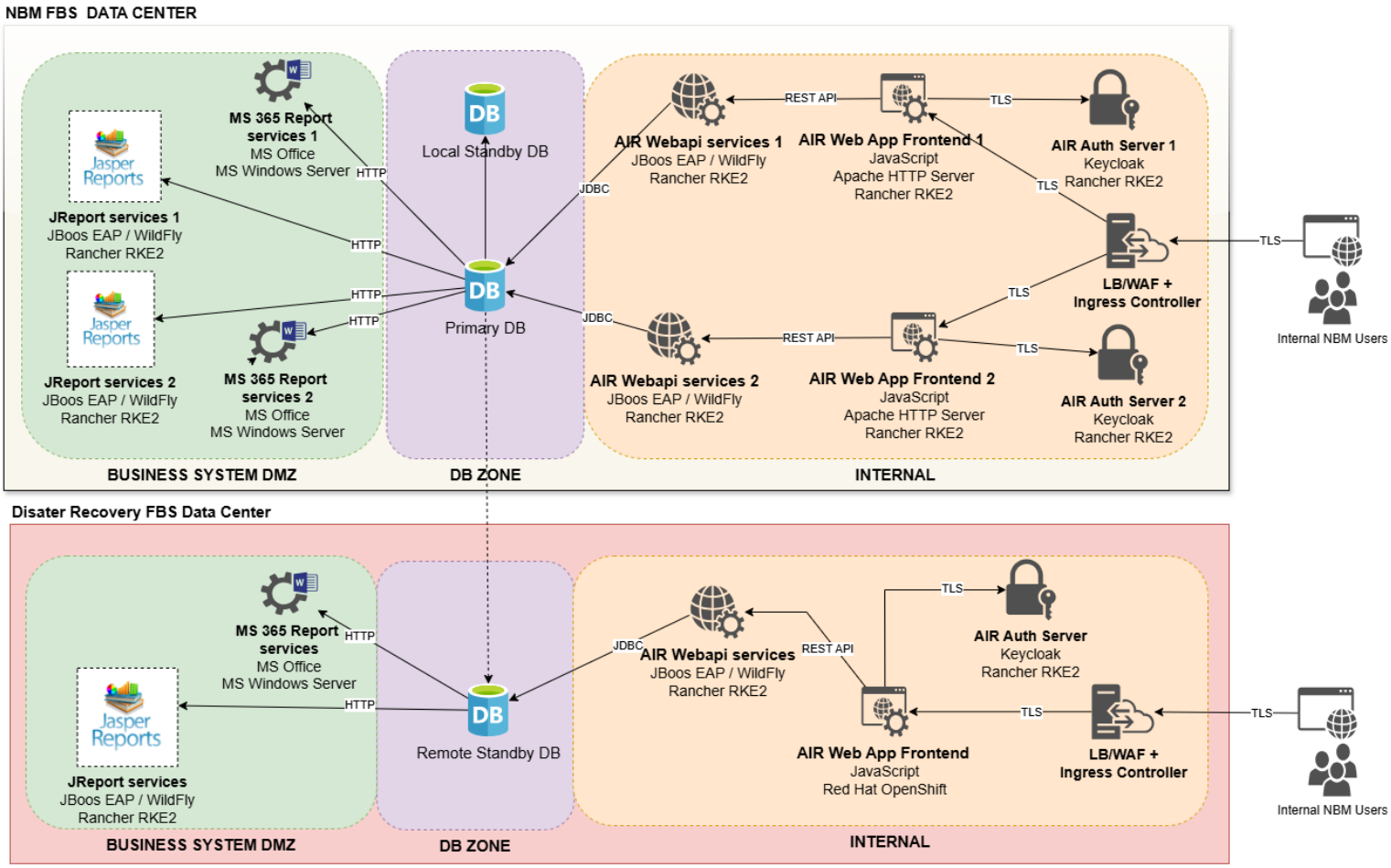


Figure 9. 3.6-1 “RKE2-based solution”

Red Hat-based solution (optional):

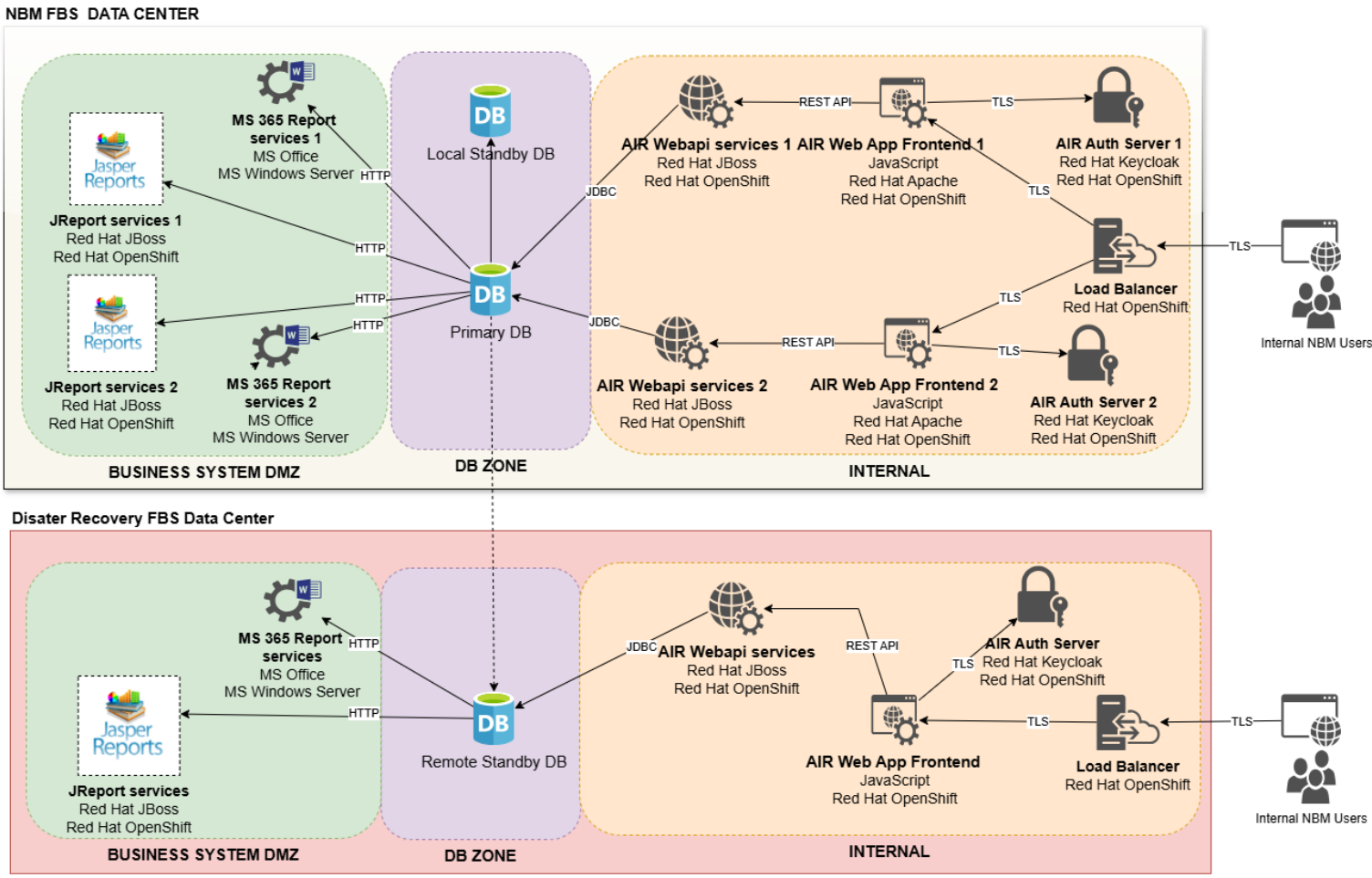


Figure 10. 3.6-2 “Red Hat-based solution”

Oracle Cloud(OCI) solution (optional):

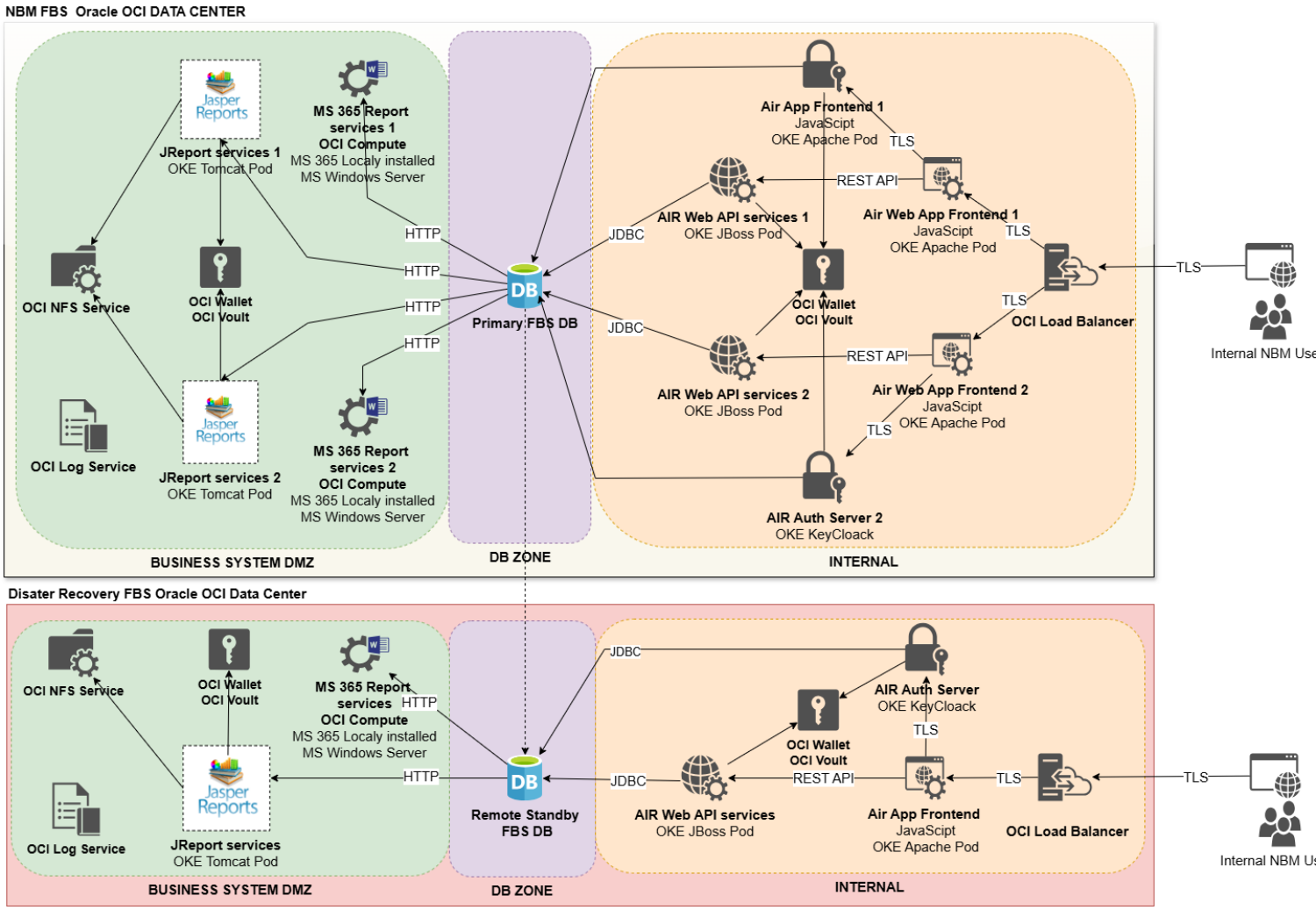


Figure 11. 3.6-3 “Oracle Cloud-based solution”

4. Layered Architecture (CNF.7)

4.1 Presentation Layer (CNF.3, CNF.13, CNF.14, CNF.46, CNF.47, CNF.48)

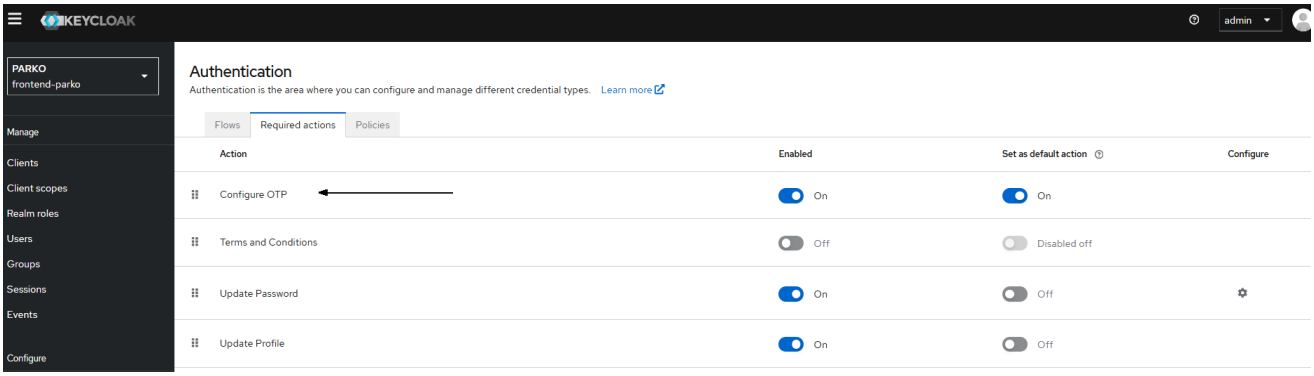
The presentation layer provides user interaction and user experience management. It does not implement authoritative banking business rules except for client-side format checks and user convenience validations. Final validation and business decisioning are executed by the business logic layer and database APIs.

The proposed solution includes two complementary user interface layers:

- **Backoffice**: implemented using Oracle Forms for operational efficiency and data-intensive workflows
- **Front/middle-office and remote banking**: implemented using modern React-based web applications for customer-facing interactions

The system is accessible to authorised users using standard workplace computing resources, including desktop workstations, laptops and virtual desktop / VDI environments. Authorised users can access the application through the standard workplace environment, authenticate successfully, perform business operations, and

print application-generated documents or reports using available workplace printers. Each authorised user can access the system using MFA-enabled authentication:



4.1.1 Compliance with GUI Design Principles (CNF.12)

Both user interface layers are designed in alignment with established usability principles, ensuring consistency, clarity, and efficiency across all user interactions:

Structure Principle

The GUI architecture is organized using clear, modular, and consistent design patterns:

- React:
  - Component-based architecture ensures reusable and predictable UI elements
  - Consistent layout structure (header, navigation, content area) across all pages
  - Standardized design system (forms, buttons, dialogs) ensures uniform behaviour
  - Responsive design enables seamless access across all mobile devices
  - By utilizing print-ready CSS, the application automatically adjusts output layouts to fit specified dimensions and orientations
- Oracle Forms:
  - Structured form layouts aligned with business processes (data entry, validation, review)
  - Logical grouping of fields and actions to reflect user workflows
  - Consistent navigation patterns across modules
  - This structured approach ensures users can easily recognize and understand interface elements regardless of context.

Simplicity Principle

The interfaces are designed to minimize user effort and streamline task execution:

- Clear and user-friendly language is used across both interfaces
- Frequently used actions are easily accessible via:
  - Quick action buttons (React UI)
  - Keyboard shortcuts and function keys (Oracle Forms)
- Default values, autofill, and validation reduce manual input

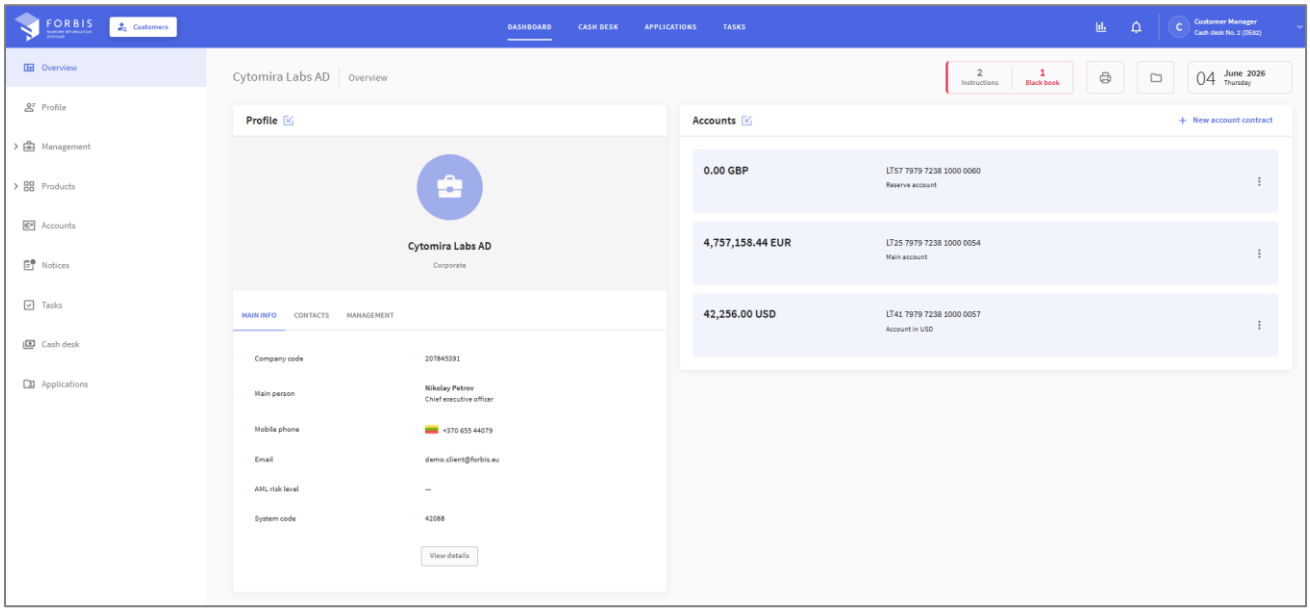
Complex workflows are broken down into smaller, manageable steps, improving usability and reducing cognitive load.

Visibility Principle

The system ensures that all necessary actions and information are visible without overwhelming the user:

- Contextual menus and dynamic rendering (React) display only relevant options
- Oracle Forms screens present only required fields based on the process stage
- Use of tabs, collapsible sections, and progressive disclosure techniques
- Clear labeling and grouping of related actions

This ensures users can focus on their tasks without unnecessary distractions.

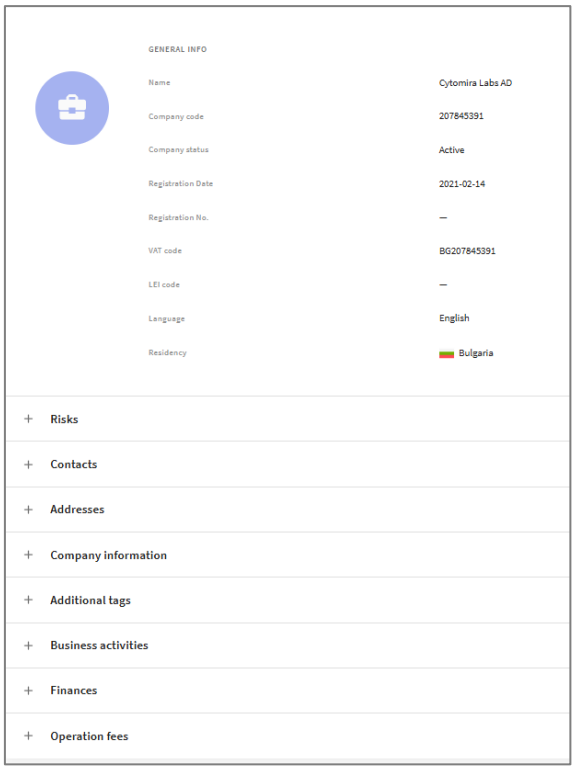


Feedback Principle

The application provides clear, timely, and meaningful feedback to users:

- Real-time validation messages (e.g., incorrect input formats)
- Status indicators for ongoing processes (loading, success, failure)
- Informative error messages using business-friendly language
- Confirmation dialogs for critical actions

Both UI layers ensure that users are always aware of system state and outcomes of their actions.



Tolerance Principle

The interfaces are designed to prevent errors and minimize their impact:

- Input validation at both client and server levels
- Confirmation prompts for irreversible actions

Oracle Forms additionally supports transactional control, allowing users to review and correct data before committing changes.

Reuse Principle

The solution emphasizes reuse of components and interaction patterns:

- React:
  - Shared component library (buttons, forms, modals)
  - Centralized styling and theming for consistency
- Oracle Forms:
  - Reusable form templates and validation logic
  - Standardized navigation and action patterns

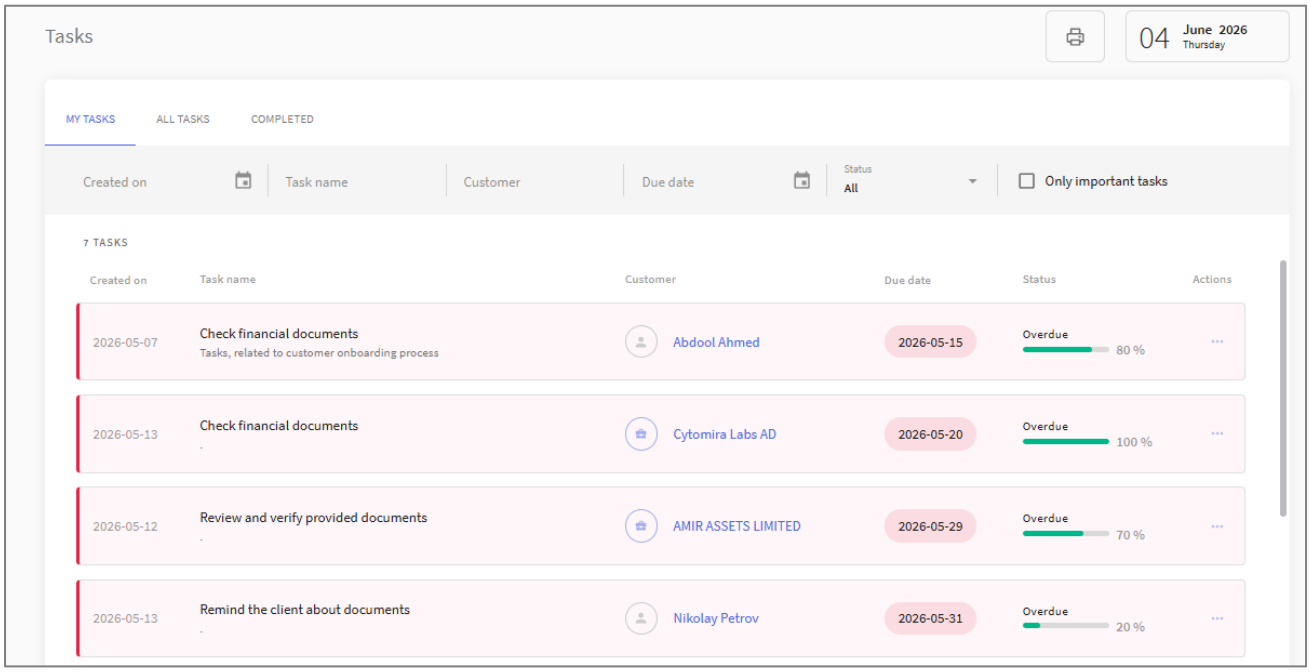
This reduces development effort and ensures a consistent user experience across the system.

Usability and Task Optimization

The GUI is specifically designed to optimize user workflows:

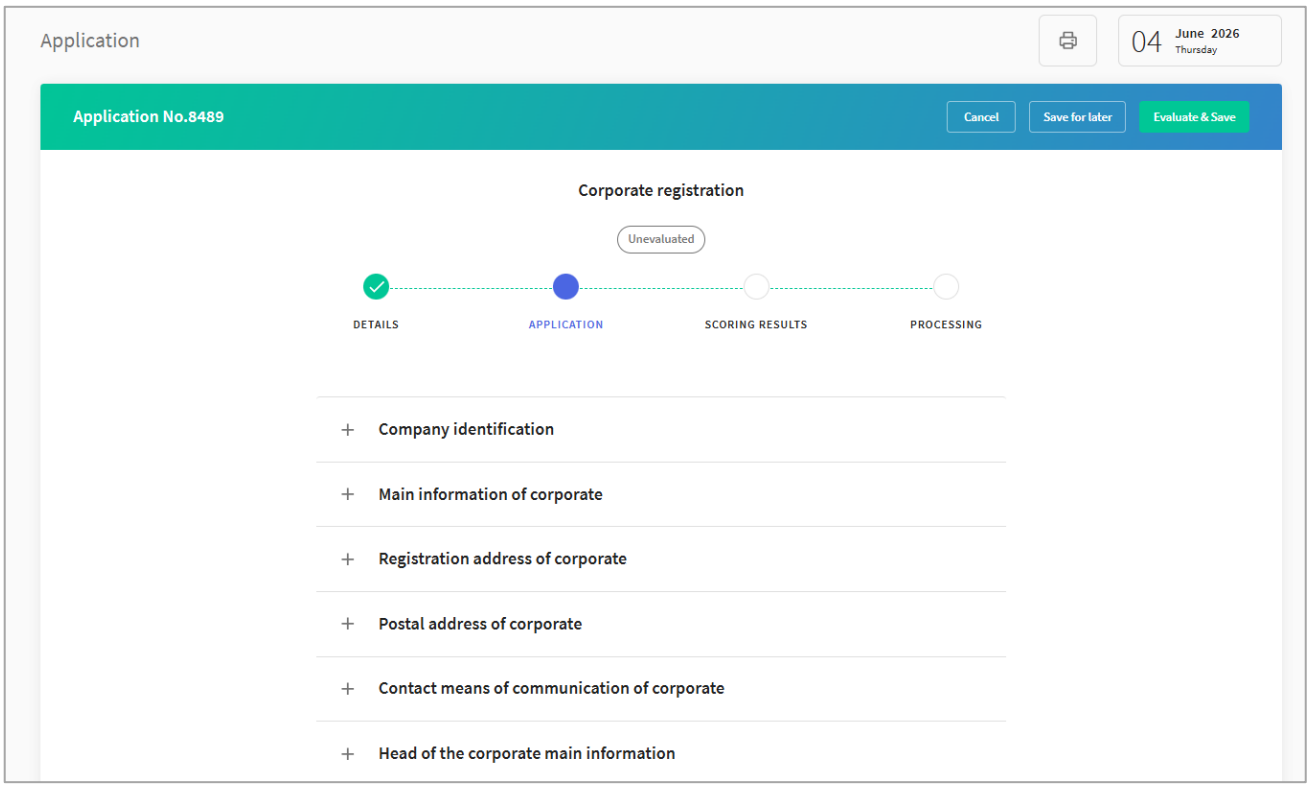
- Complex processes are simplified using:
  - Step-by-step guided interfaces (React)
  - Predefined data entry flows (Oracle Forms)
- Search, filtering, and sorting capabilities improve data accessibility
- Context-aware actions reduce unnecessary navigation

The web presentation channels are accessible through modern browsers without installing client-side business application software. The technology stack specifies support for recent versions of Microsoft Edge, Google Chrome, Mozilla Firefox and Apple Safari for React-based front-office, middle-office and internet banking applications. Back-office Oracle Forms access is provided for administrative users according to the controlled internal workstation and Java/Web Start requirements defined for that environment.



4.1.2 Client Operating Environment Compatibility (CNF.49)

The client applications supports operation in Microsoft Windows 10, Windows 11, and newer Microsoft Windows operating environments. The applications can be launched and used on standard workplace devices running supported Windows versions, including desktop workstations, laptops, and virtual desktop environments. Compatibility is ensured through the use of standard Windows-supported client technologies and has been verified by executing typical user operations, including user authentication, access to applications modules, data entry, data retrieval, and report generation.



4.2 Business Logic Layer (CNF.16, CNF.17, CNF.18, CNF.19, CNF.20, CNF.21, CNF.22, CNF.23, CNF.24, CNF.50)

Business logic is implemented centrally through Oracle database APIs and procedures, exposed to applications services through controlled interfaces. This enables consistent validation across back-office, front-office and middle-office, remote banking and integration flows. Java/Web API services act as controlled access points to the database business logic and support service-oriented decomposition for channels and integrations.

- Business workflows orchestrate multi-step banking processes and route work through the required validations and authorizations.
- Business entities encapsulate rules and data consistency for accounts, customers, documents, transactions, products and reference data.
- External systems do not access private database objects directly; they use defined applications interfaces or integration platform(ESB) services.
- Asynchronous operations support working-day processing, end-of-day processes, integration retries and scheduled jobs.

Independence from Presentation Layer

The Business Logic Layer is fully decoupled from the user interface:

- React applications interact exclusively via RESTful APIs, without embedding business rules in the frontend
- Oracle Forms interfaces invoke database-stored procedures and APIs, ensuring that business logic is not hardcoded in the UI triggers
- No duplication of business rules across UI technologies

This guarantees that changes in the presentation layer (e.g., UI redesign or technology replacement) do not impact core business logic.

Independence from External Systems

The Business Logic Layer exposes well-defined, technology-agnostic interfaces:

- Service endpoints (REST/JSON) act as contract-based interfaces
- External systems integrate through these interfaces without direct access to internal logic or database structures

This allows independent evolution of internal logic without breaking external integrations.

Business entities are derived from business domain analysis and reflect key concepts within the system. These entities are consistently defined and used across all layers of the application. Examples of identified business entities include Customer, Contract, Transaction, etc. Each entity is uniquely defined with clearly specified attributes, relationshFBS, and lifecycle states.

All business entities are encapsulated within dedicated components responsible for managing their data and rules.

Key characteristics:

- Each entity has a single source of truth within the Business Logic Layer
- All validations, constraints, and business rules related to the entity are centralized
- Direct access to underlying data structures is restricted

Implementation:

- PL/SQL packages representing each entity or domain group
- Controlled access via procedures/functions (APIs)
- Database constraints and triggers supporting data integrity

The related Business logic layer components communicate with each other through dedicated internal interfaces / functions implemented as PL/SQL packages or RESTfull API.

4.2.1 Integration interfaces and Extensibility (CNF.51)

The diagram bellow illustrates the proposed integration and extensibility concept. The Solution exposes and consumes business services through standardized service/API interfaces and uses the Integration Gateway / Integration Platform to connect with existing and future NBM systems.

The architecture supports business-function integration, structured data exchange, security integration, and future extensibility through open and well-defined interfaces. New NBM components or modules can be connected without changes to the core application, using standard APIs, agreed data formats,

authentication/authorization mechanisms, and integration contracts. This approach enables controlled interoperability with NBM applications while preserving modularity, maintainability, and future scalability of the Solution.

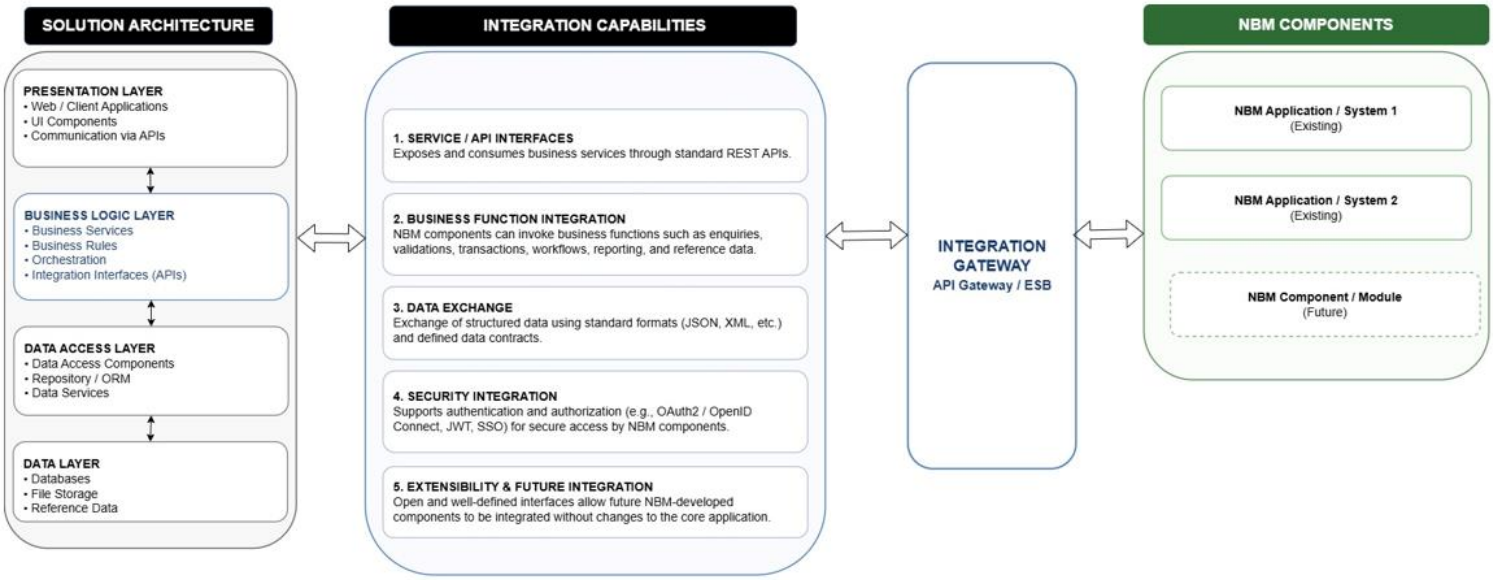
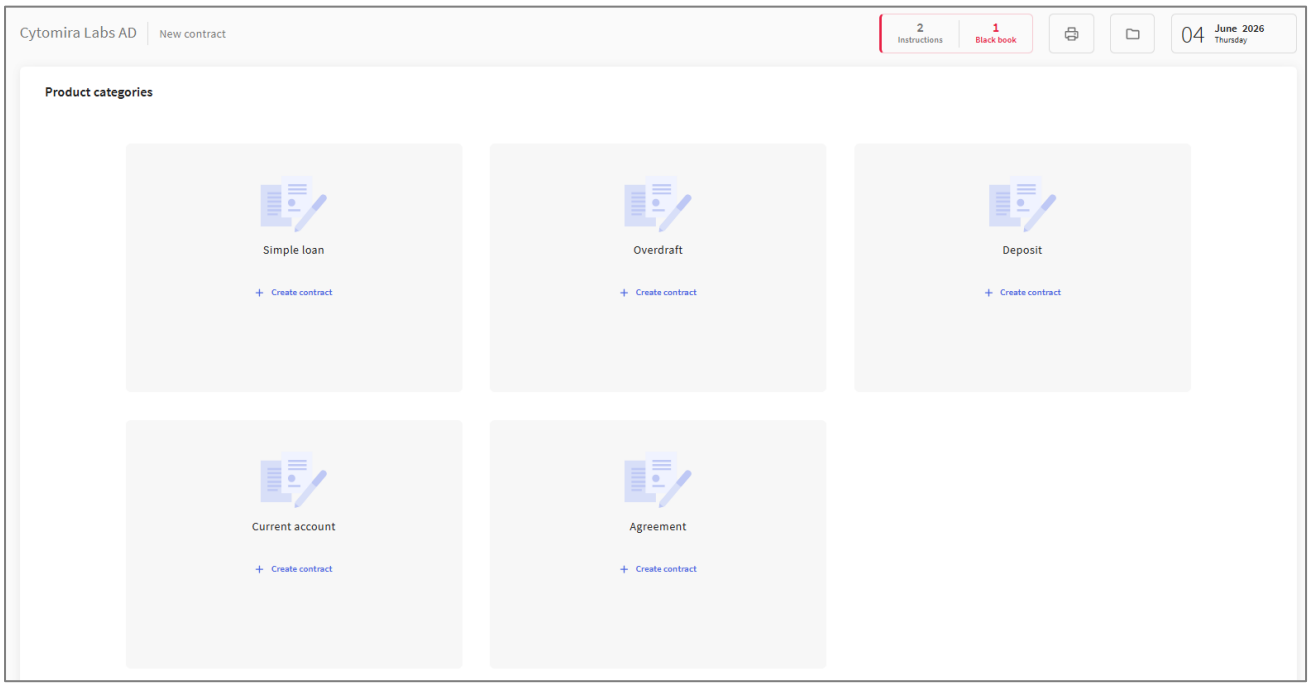


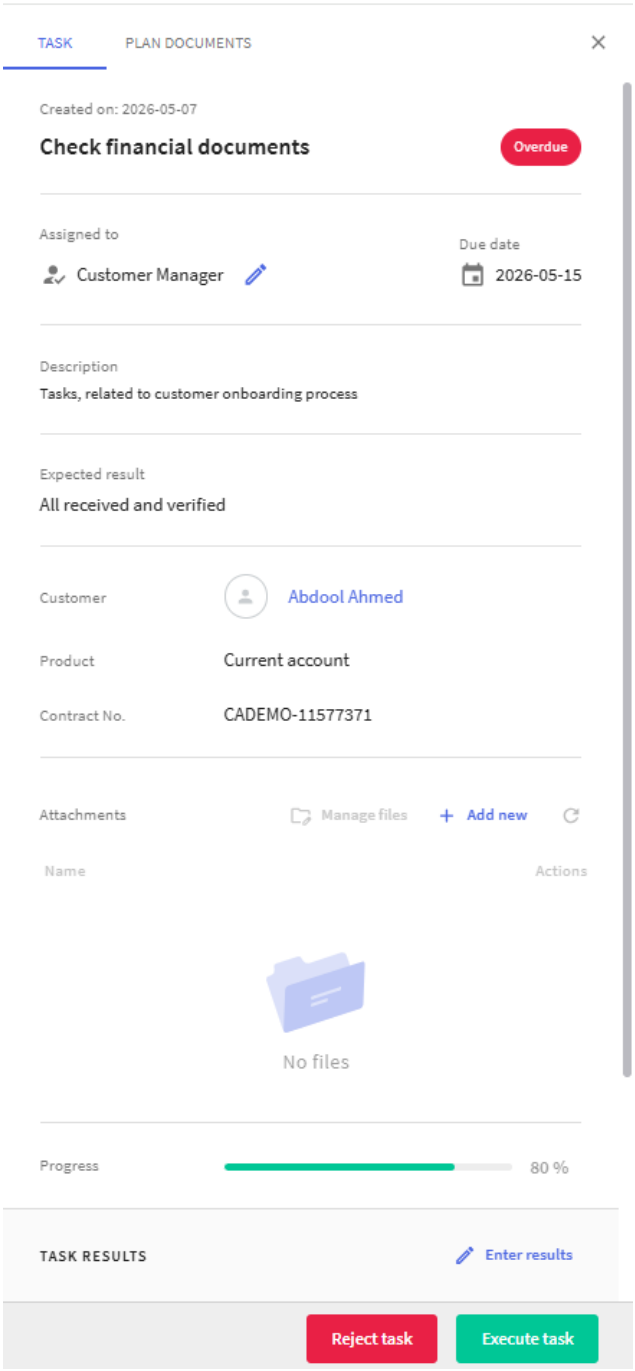
Figure 12. 4.2.1-1 “Extensibility of proposed Solution”

4.2.2 Administrative functions for system flexibility (CNF.90, CNF.91, CNF.97, CNF.141, CNF.180)

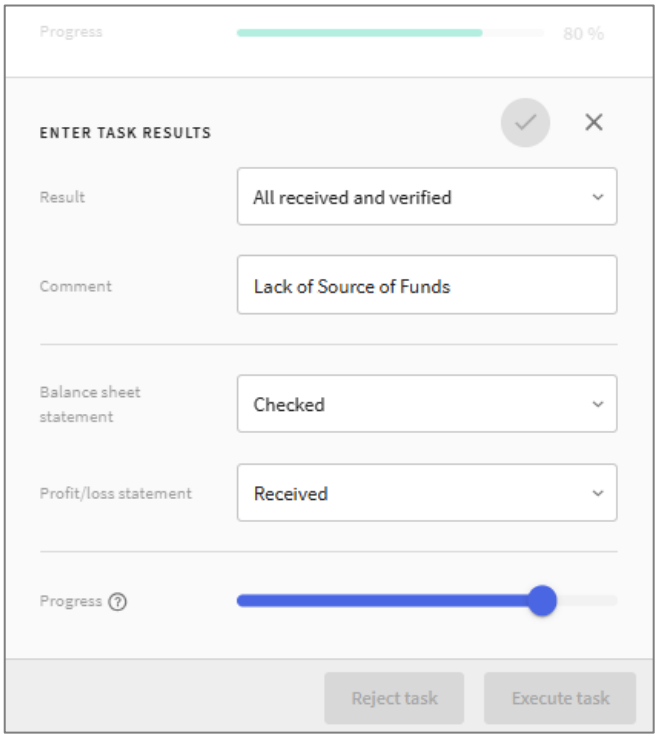
The system allow for administrators to customize applications functionality according to business needs. Banking products functionality adapt to NBM business requirements .



Internal tasks' management processes .

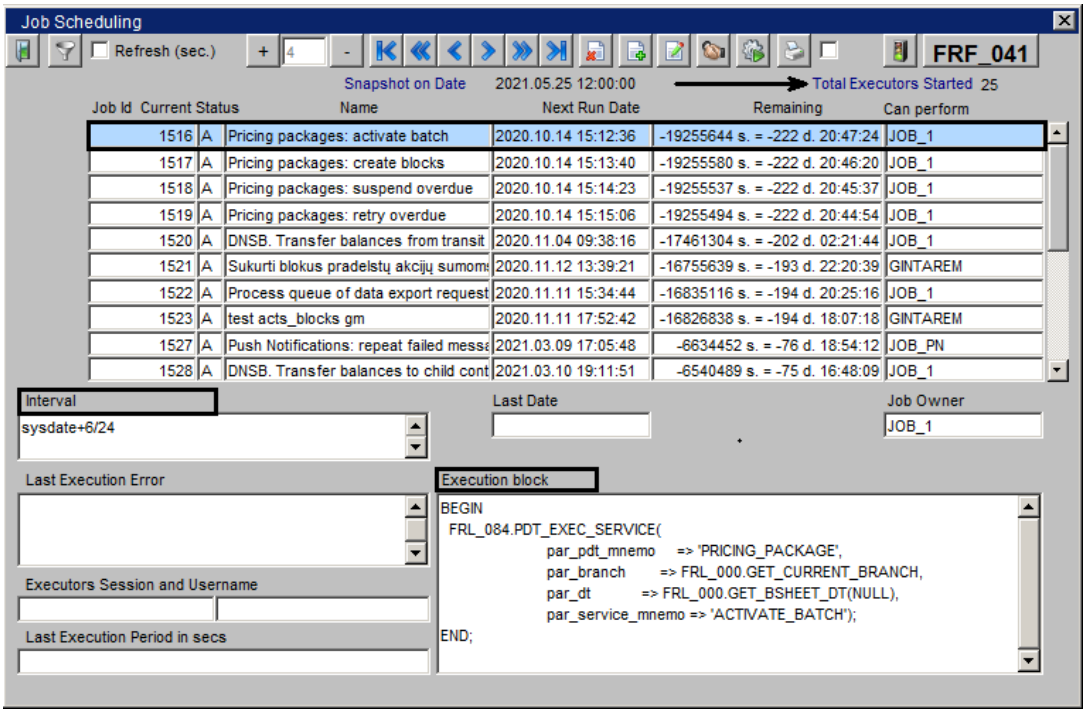


From the bank employee view, tasks and plans management is straightforward and user friendly.

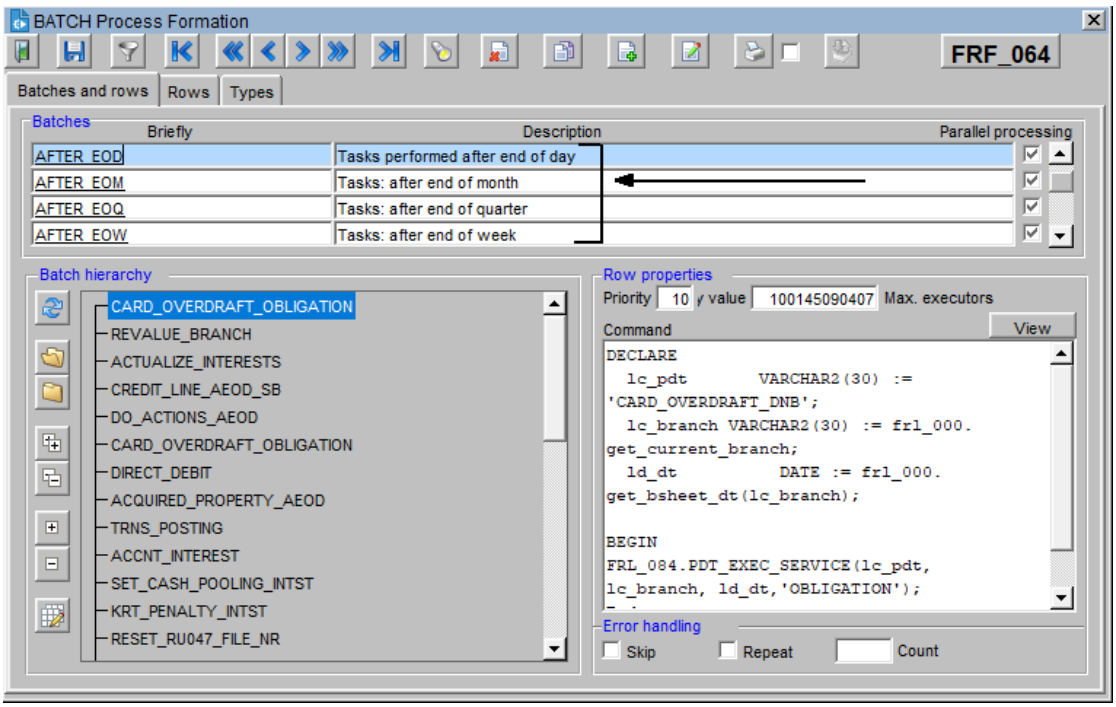


Technical personnel, for example system administrators/advanced users, would need to configure the tasks in the back office, workspace for advanced uses.

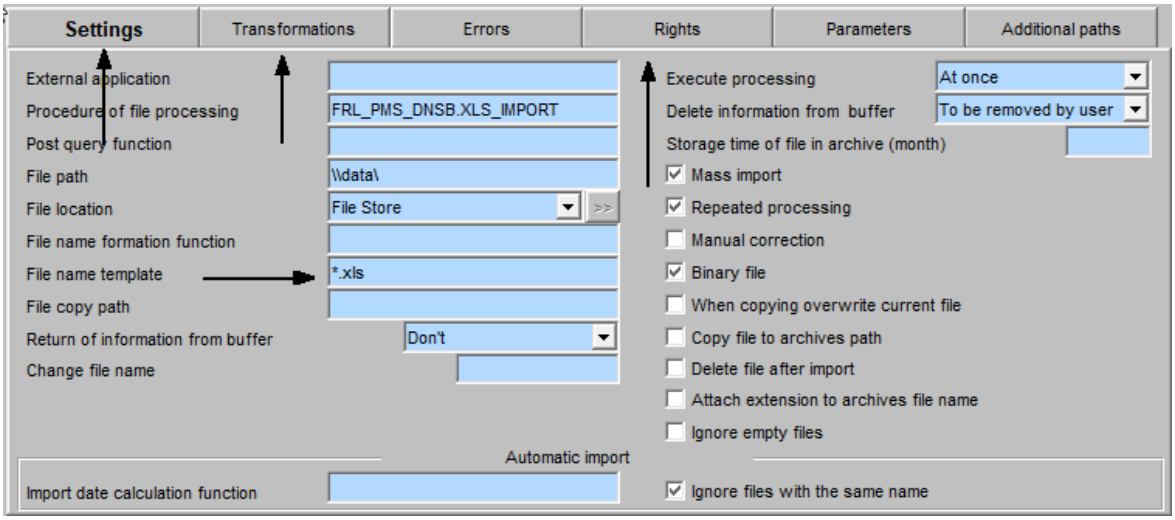
Execution of the system's jobs(scheduled processes). The system shall provide the capability to create, modify, and terminate existing automatic system processes, as well as to change their execution frequency:



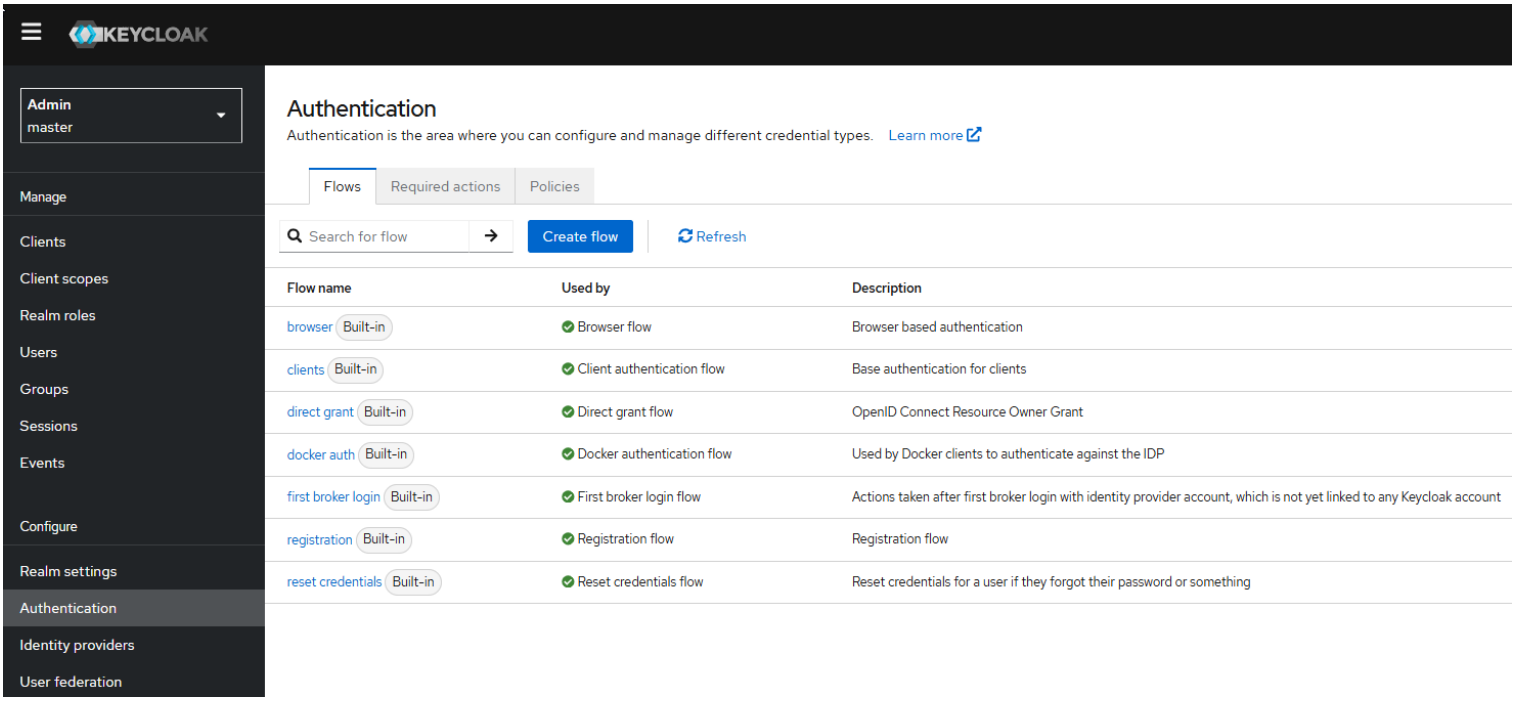
- DAY/MONTH/YEAR closing processes:



Also there are enhancement and management of a wide range of system classifiers. For example, internal classifiers, there is possibility to import classifier from external systems. There is possibility to manage imported/exported data formats and apply transformation.



Manage authentication and authorization process:



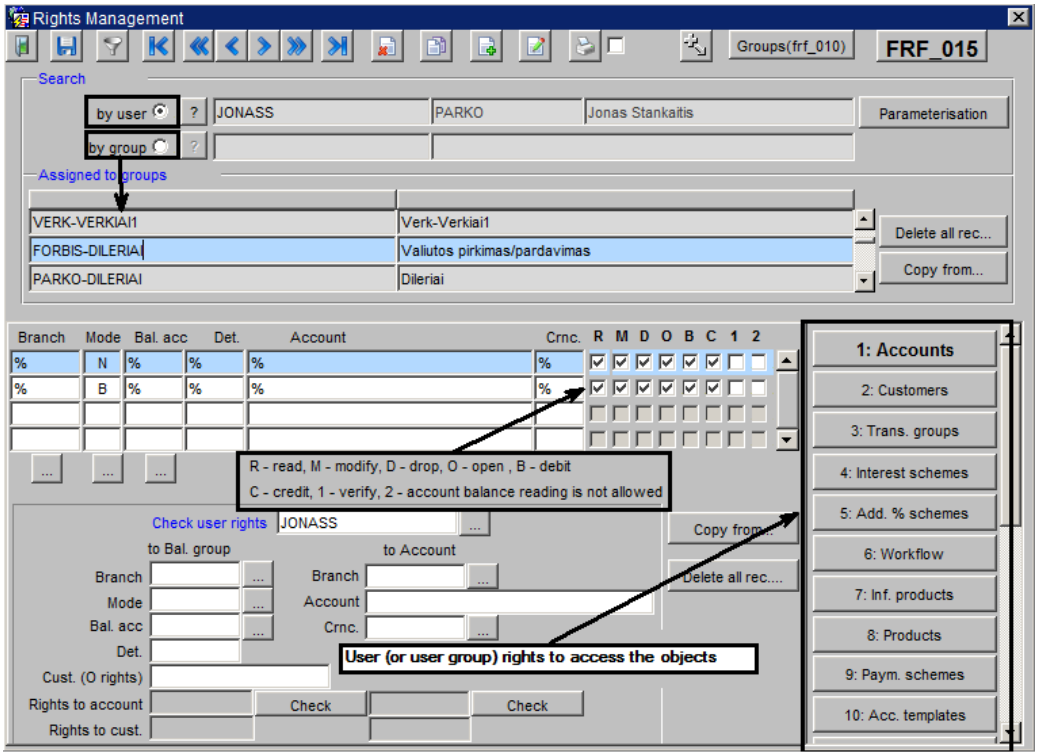
4.2.3 Administrative functions for system management (CNF.90, CNF.155, CNF.156, CNF.157, CNF.158, CNF.159, CNF.160, CNF.162, CNF.166)

The system shall provide system administrators with additional capabilities to manage and control the system in such scenarios:

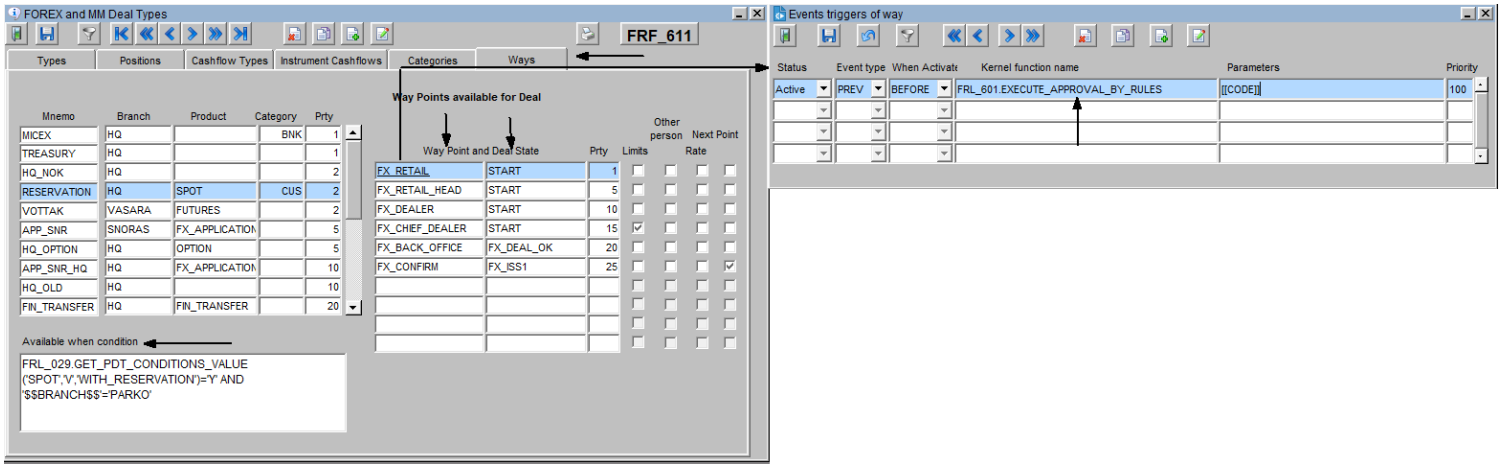
- (CNF.141) The system is optimized for operating with a large number of users. That is why instead of setting rights and roles for each particular user, the rights and roles are defined for groups of users.

The system supports unlimited number of user groups, and the groups are created according to the business needs. In fact, a user group is a set of users having the same business role. A user group may have different

access rights to various groups of the FBS objects (accounts, customers, products, payment messages, etc.). Once the user has been included into the group, he/she obtains all the privileges the group provides.



Possible to manage validation rules, limits, approval rules of applications, notifications, statuses, transitions, approvals:



And many other parameters such as:

1. **System user settings**:

1.1. User personal information, including the definition of the permissions with which the user may access the system and the system component interfaces to which the user is allowed to connect.

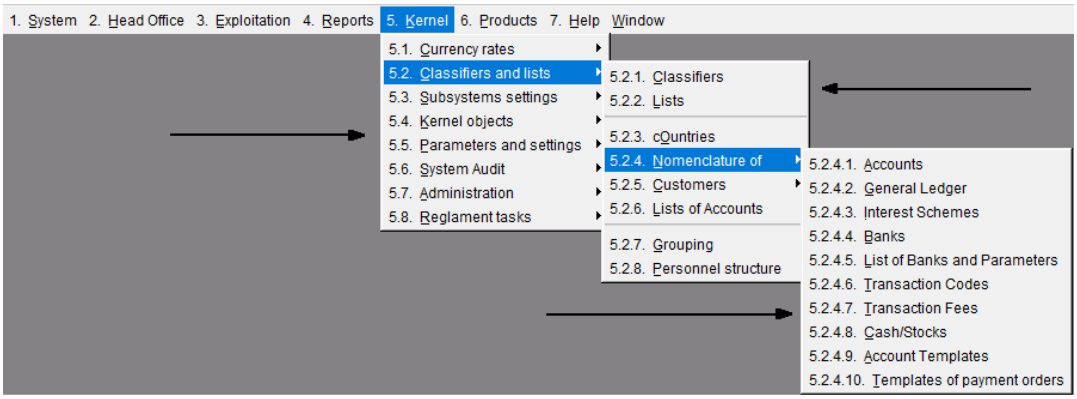
1.2. Assignment of users to specific database roles and network access control.

1.3. Granting and changing user passwords, as well as blocking and unblocking users.

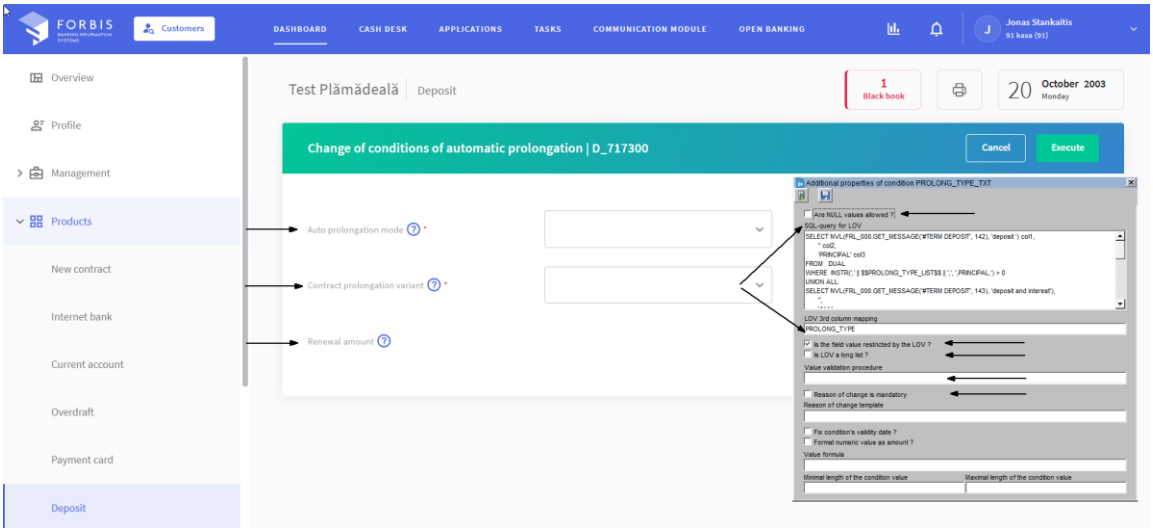
2. **User group settings**:

2.1. Definition of access rights to specific business functions of the applications layer, including access to accounts, clients, transactions, payment information, etc.
2.2. Definition of access rights to specific menu items.
2.3. Definition of access rights to specific user interface fields, buttons, and other elements.

- **Assignment of system users to user groups.**
- **Monitoring of system user sessions**.
  - The system shall provide the capability to view all user sessions and, where necessary, terminate them or enable action tracing for a specific session.
- **Maintenance of system encryption keys.**
- **System registry value settings.**
  - The system shall provide the capability to define system registry values that control the operation of system business functions and to assign individual values to a specific user.
- **System data import/export.**
  - The system shall provide the capability to transfer system settings and applications layer data from other databases, including FORBIS_DEV, TEST, and DEV.
- **Creation of system users and their passwords**.

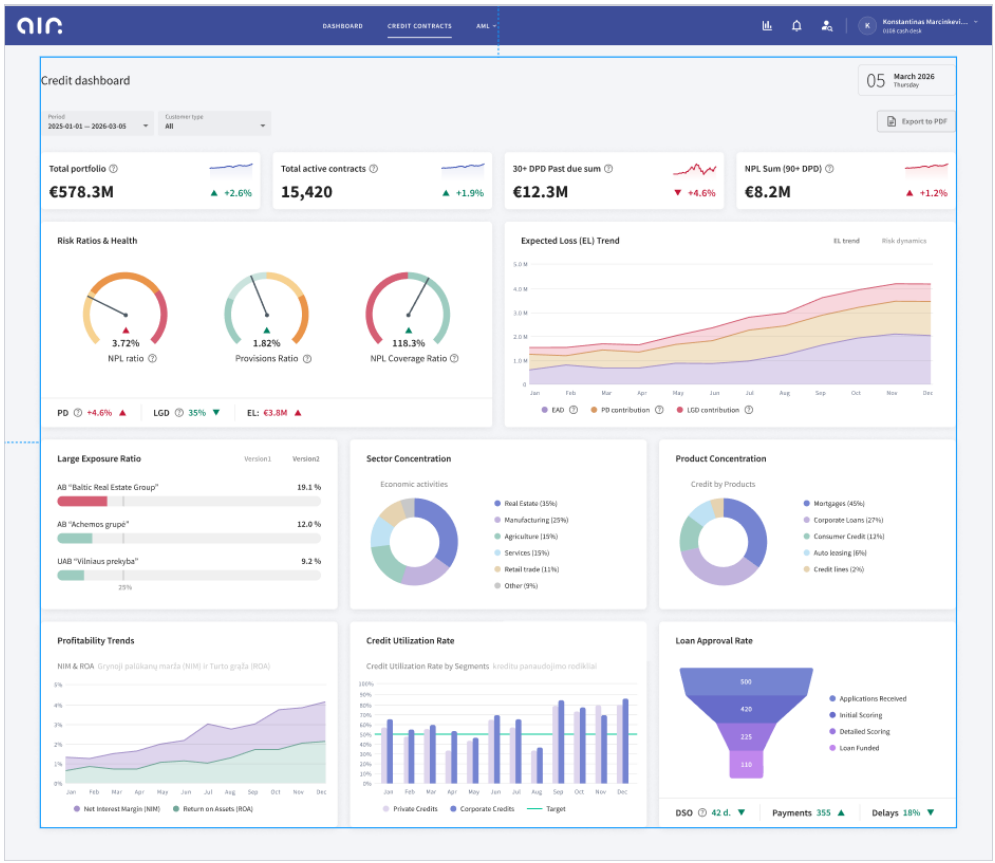


The system provides the capability to manage the user interfaces for internal business users, process management users, remote banking users, and reporting users. The same configuration settings shall enable administrators to manage screens across multiple UI modules. Each administrator action is fixed in audit storage.

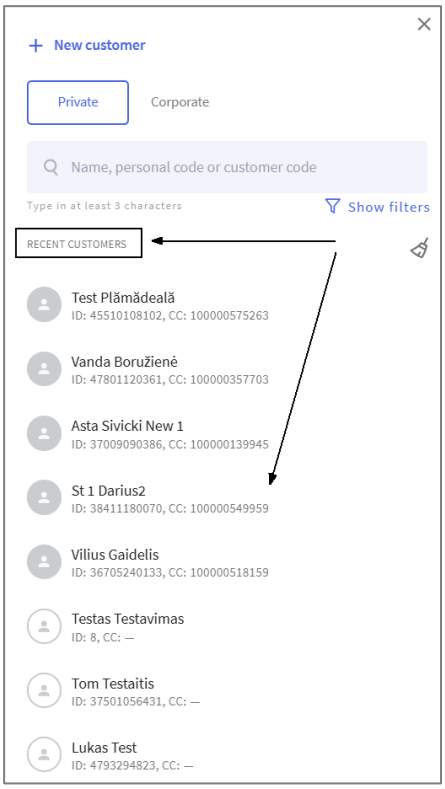


4.2.4 Users' functions for system flexibility (CNF.130)

NBM business users can manage dashboard views:



and access frequently used functions, use personalized navigation options where available:



4.3 Data Access Layer (CNF.26, CNF.27, CNF.28, CNF.30, CNF.31, CNF.32, CNF.33, CNF.34, CNF.52, CNF.53, CNF.132, CNF.133, CNF.190)

The data layer is based on Oracle Database 19c Enterprise Edition. It is the single authoritative data repository for core banking data and related subsystem data. Database APIs enforce data uniqueness, mandatory data, transaction consistency, referential integrity, validation and auditability. JDBC is used by Java-based services to access controlled database procedures and APIs.

- Oracle RAC and local standby mechanisms support local resilience.
- Oracle Active Data Guard supports the remote standby database for disaster recovery.
- RMAN backups, Flashback options and controlled archiving support recovery and data lifecycle management.
- UTF-8/Unicode support enables multilingual data handling and banking reporting needs.

The database is divided into dedicated logical schemas - Core, Remote Banking, Front Office, Middle Office, Integration and Remote Service. Each schema is responsible for storing the relevant data entities, business rules, and supporting logic for its functional area. This approach allows the system to maintain a single source of truth while keeping business domains well organized and clearly separated.

By centralizing the data layer, the solution ensures that all modules operate on consistent and reliable data. Business logic is maintained close to the data, enabling efficient processing, reduced duplication, improved

data integrity, and simplified governance. At the same time, the schema-based separation supports modular development, easier maintenance, controlled access, and future extensibility.

This architecture provides a robust foundation for enterprise-level operations, ensuring that data and business logic are managed in a secure, structured, and transparent manner across all functional areas of the platform.

The database schema is designed in accordance with standard normalization principles:

- The model adheres to Third Normal Form (3NF), and where applicable, higher normalization levels.
- Data is decomposed into logical entities to eliminate duplication and redundancy.
- Each table represents a single business concept with clearly defined attributes.

This ensures efficient data storage, reduced risk of data anomalies (insert, update, delete) and improved data consistency.

Redundant data storage is avoided through:

- Use of reference tables and master data entities.
- Proper decomposition of repeating groups into separate relational tables.
- Avoidance of duplicated attributes across entities.

Where denormalization is considered (e.g., for performance optimization), it is applied selectively and controlled, ensuring that data consistency is not compromised. All relationshFBS between data entities are explicitly defined and enforced at the database level: primary keys, foreign keys, unique constraints, check and not null constraints. These constraints ensure that the data remains consistent, accurate, and aligned with business rules. The data model is derived directly from business requirements and domain analysis:

- Each entity corresponds to a clearly defined business concept (e.g., Customer, Contract, Transaction).
- RelationshFBS reflect real-world business interactions and dependencies.
- Attribute definitions are aligned with business meaning and usage.

This ensures that the data model accurately represents the business domain and supports all required processes. The application supports an integrated data model for reference information by centralizing all common nomenclatures (e.g., countries, currencies, statuses, product types) within the Data Layer.

All CRUD operations are encapsulated within business logic components, which enforce validation, authorization, and business rules.

The database schema is designed based on business domain concepts, without embedding process-specific or application-specific logic. Data structures (tables, relationshFBS, constraints) are defined independently of how business workflows are implemented. No procedural or workflow logic is stored within the data model, instead, such logic is handled exclusively in the Business Logic Layer. Standardized data representations enable reuse across multiple services, processes, and external integrations.

The data architecture is designed to efficiently support transactional processing (OLTP). Analytical workloads are supported via separate reporting structures (e.g., materialized views, data marts, or replicated schemas) and denormalized models to improve query performance.

Technical Documentation of data layer includes description of database structures (tables, columns and relationshFBS), database objects (views) and Entity-Relationship (ER) diagrams illustrating data structure and dependencies. Relevant semantic metadata is exposed within the application (e.g., report configuration, field descriptions, tooltFBS). The Oracle database leverages ACID-compliant transactions and row-level locking to

prevent data conflicts. Appropriate isolation levels ensure consistency while maintaining performance. All data modifications are performed through the Business Logic Layer, which enforces validation and business rules. Database constraints (PK, FK, CHECK, UNIQUE) guarantee structural integrity. Mechanisms such as optimistic locking (versioning/timestamps) and transaction control prevent overwriting concurrent changes. In case of conflicts, transactions are safely rolled back or retried. Users are clearly informed of conflicts, errors, or concurrent modifications through meaningful messages. UI components (Oracle Forms and React) display real-time feedback, allowing users to take corrective actions.

4.3.1 Backup and Historical Backup Management Capabilities

The proposed solution fully meets and exceeds the stated requirement by leveraging Oracle's native, enterprise-grade backup and recovery ecosystem, primarily Oracle Recovery Manager (RMAN), combined with Oracle Enterprise Manager (OEM) and supporting database features.

The solution supports fully automated backup execution using Oracle-native mechanisms:

- Oracle Recovery Manager (RMAN) provides scripting capabilities for defining backup strategies (full, incremental, archived redo log backups).
- Oracle Scheduler (DBMS_SCHEDULER) or external enterprise schedulers can be used to configure and automate backup jobs with flexible frequency (hourly, daily, weekly, or custom intervals).
- Integration with Oracle Enterprise Manager (OEM) enables centralized scheduling, job orchestration, and policy-based automation across environments.

This ensures that backup operations can be configured to align with RPO/RTO requirements and operational constraints.

5. High-Availability and Resource-Efficiency Architecture (CNF.36, CNF.37, CNF.38)

The solution is deployed on a fully virtualized infrastructure and operates as a set of stateless services orchestrated by Kubernetes. This platform ensures continuous availability of all application components by distributing workloads across multiple nodes and automatically maintaining service health. Kubernetes performs ongoing health checks, restarts failed containers, and provides node-level failover, ensuring that application services remain accessible even during hardware disruptions or localized failures.

To eliminate single points of failure, all critical components are deployed in redundant configurations. Application services run as multiple instances behind load balancers, allowing traffic to be routed only to healthy nodes. The messaging layer uses replicated ActiveMQ Artemis brokers, ensuring that message delivery continues uninterrupted even if one broker instance becomes unavailable. Because each component can fail independently without affecting the overall system, the architecture maintains operational continuity under a wide range of failure scenarios.

Processing resources are managed efficiently through a combination of application-level tuning and container-level resource controls. Kubernetes enforces CPU and memory limits for each service, ensuring balanced resource consumption across the cluster. Stateless service design and horizontal scaling allow the system to adapt to varying workloads, while resource isolation prevents individual components from degrading the performance of others. This approach ensures rational, predictable, and balanced use of processing capacity across the entire platform.

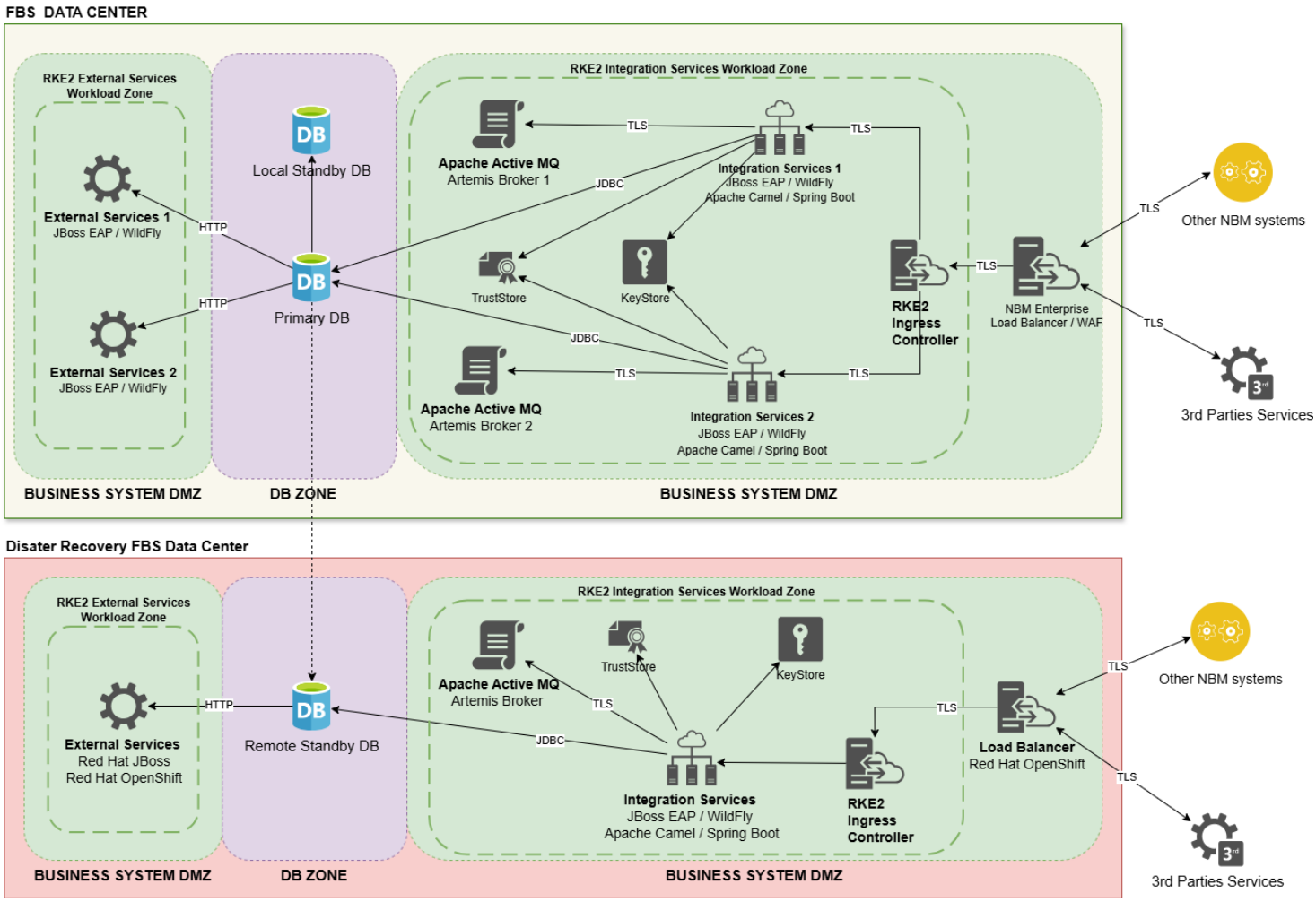
6. Integration Architecture (CNF.10, CNF.40, CNF.41, CNF.42, CNF.43, CNF.44, CNF.59)

The integration layer is designed to maintain minimal dependence on any specific underlying technology platform, ensuring long-term scalability, flexibility, and ease of maintenance. All components are implemented as **Spring Boot microservices packaged in Docker containers**, which allows them to run consistently across different operating systems and cloud environments. By relying on Kubernetes as the orchestration layer, the platform benefits from a standardized runtime environment that abstracts away infrastructure differences and enables seamless deployment on virtualized servers or private cloud platforms.
The technological stack is intentionally built on **open, widely adopted, and vendor-neutral technologies**. Spring Boot provides the service runtime, Apache Camel delivers routing and integration capabilities, ActiveMQ Artemis ensures reliable messaging, Docker handles containerization, and Kubernetes manages orchestration and scaling. These technologies follow open standards, have broad industry support, and avoid any dependency on proprietary supplier-specific components, ensuring long-term maintainability and interoperability.
All application components are **fully hardware-agnostic**. No component relies on specialized hardware, and the containerized deployment model ensures consistent behaviour across development, testing, and production environments.
The architecture is explicitly optimized for **cloud-native operation**, particularly in private cloud environments. Stateless microservices enable horizontal scaling, while Kubernetes provides automated failover, self-healing, and resource orchestration. Lightweight Spring Boot services minimize latency, and container isolation ensures that failures are contained at the service level. Parallel processing is achieved through independent microservice instances, allowing workloads to be distributed efficiently across the cluster. Kubernetes resource limits and scheduling policies further optimize CPU and memory utilization, ensuring predictable performance under varying load conditions.
To maintain simplicity and operational consistency, the platform uses **a homogeneous technology stack** across all integration components. Java, Spring Boot, Apache Camel, Docker, and Kubernetes form the unified foundation for development, deployment, monitoring, and maintenance. This reduces the number of technologies that administrators must manage, simplifies troubleshooting, and ensures that operational practices remain consistent across the entire system.

Integration is supported through the following mechanisms:

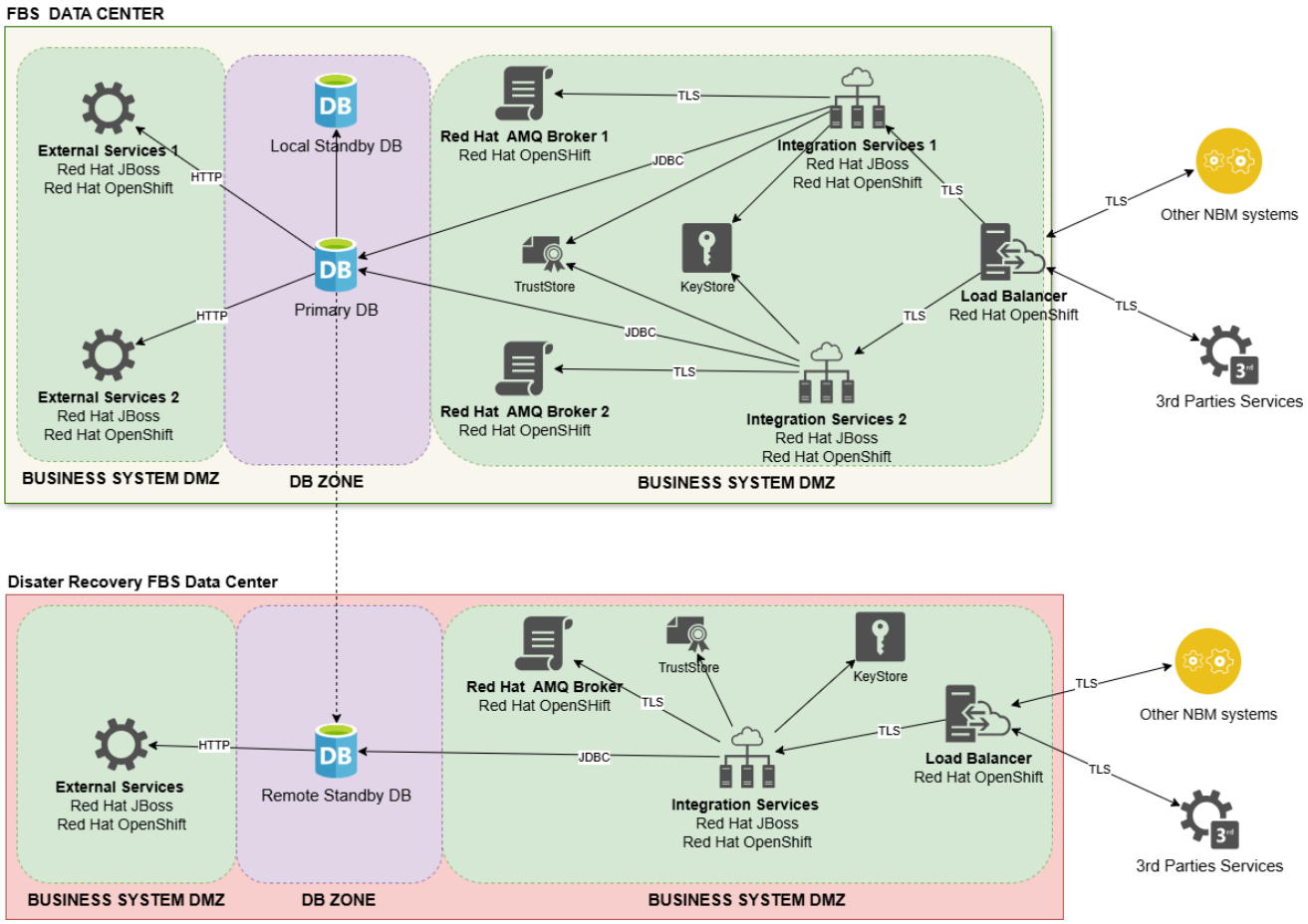
- **Synchronous integration** is provided through secured REST/SOAP/HTTPS and TLS-based service communication where immediate request-response processing is required.
- **Asynchronous integration** is supported through Red Hat AMQ Broker and integration service retry logic, enabling reliable message delivery and decoupling between systems.
- **Transformation and orchestration** are implemented through integration services and Apache Camel routes, supporting various message formats.
- **File-based and batch integration** is supported through SFTP and controlled file exchange mechanisms where required by legacy systems or batch-oriented interfaces.
- **Database and applications integration** is supported through controlled JDBC connectivity between integration services and the applications database layer, where applicable.
- **External service integration** is supported through secured load-balanced access to Microsoft 365, third-party services, and other external platforms.
- **Security controls** include TLS, trust stores, key stores, authentication, authorization, controlled DMZ network placement, and audit logging.
- **Traceability and monitoring** are supported through correlation identifiers, allowing integration-layer logs to be linked with applications, database, and business logs for end-to-end tracking.

RKE2-based solution (suggested):



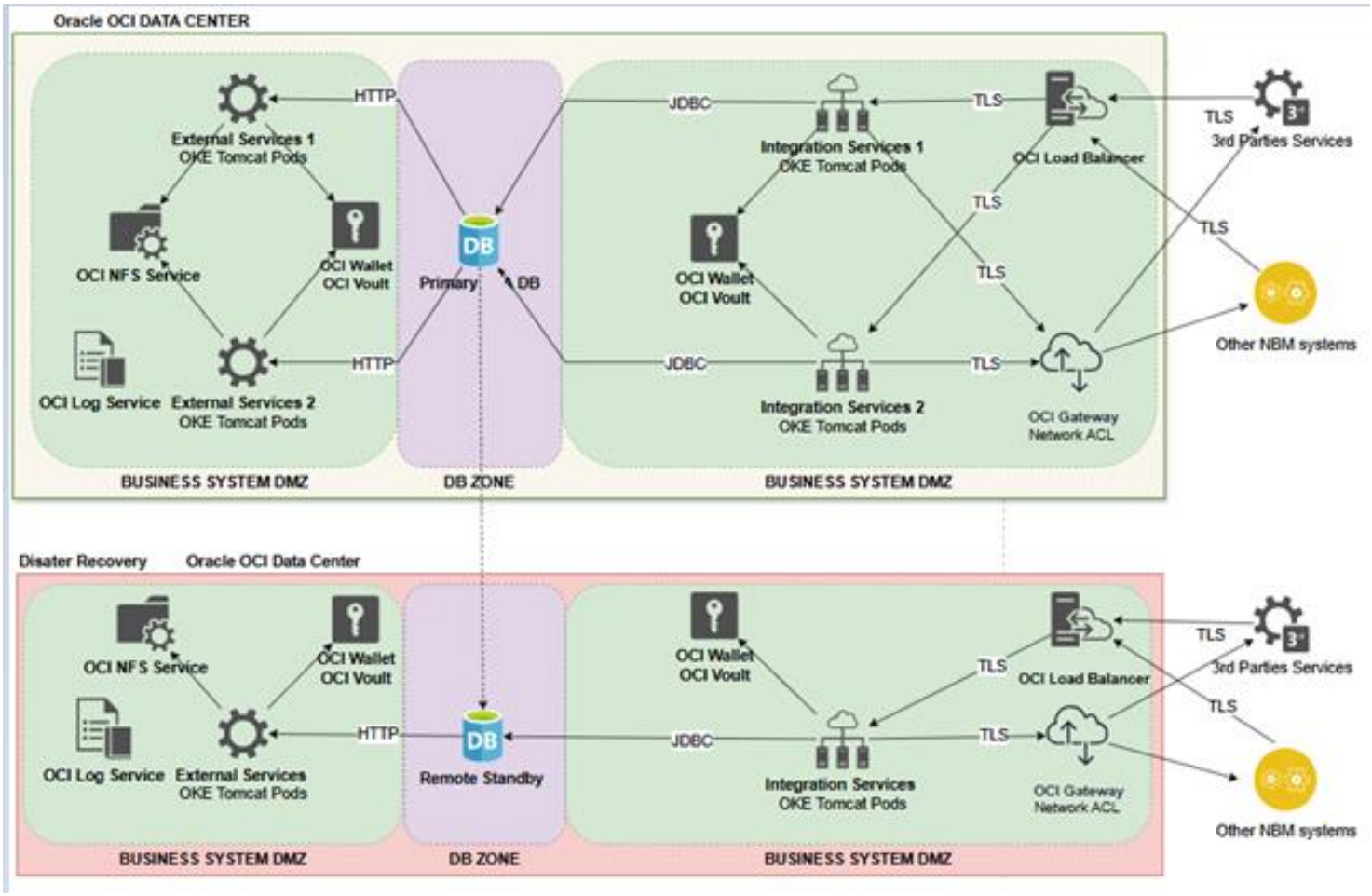
Picture 6-1 “RKE2-based solution”

Red Hat-based solution:



Picture 6-2 “Red Hat-based solution”

Oracle Cloud Infrastructure (OCI)-based solution:



Picture 6-3 “Oracle Cloud-based solution”

6.1 Enterprise Service Bus (ESB) (CNF.60, CNF.61, CNF.62, CNF.63, CNF.64, CNF.65, CNF.66, CNF.67, CNF.68, CNF.69, CNF.70, CNF.71, CNF.72)

The ESB is implemented as a distributed integration runtime composed of independently deployable Java microservices. Each ESB module is built on Spring Boot and Apache Camel, providing a uniform execution environment for routing, transformation, protocol mediation, and orchestration. The ESB operates without a central monolithic broker; instead, each integration flow is encapsulated in its own service, enabling isolated deployment, scaling, and fault containment.

The ESB operates without a central monolithic broker, instead, each integration flow is encapsulated in its own service, enabling isolated deployment, scaling, and fault containment. The Supplier’s proposed ESB architecture remains platform-agnostic and technology-flexible, allowing deployment on enterprise-grade Java runtime, container, or middleware platforms aligned with the Bank’s technology standards, operational requirements, and infrastructure strategy.

6.1.1 Runtime Architecture

Each ESB service runs as a containerized process. The runtime stack includes:

- **Spring Boot** for service lifecycle, configuration, and HTTP endpoints

- **Apache Camel** as the routing engine
- **ActiveMQ Artemis** for asynchronous messaging
- **Internal shared libraries** for logging, security, error handling, and telemetry

All services are stateless; state is externalized to messaging queues or downstream systems.

6.1.2 Apache Camel

Apache Camel provides the core routing engine. Camel routes implement Enterprise Integration Patterns:

- Content-based routing
- Message transformation (XML, JSON, binary payloads)
- Message enrichment and normalization
- Multi-step orchestration across multiple systems
- Others

Apache Camel supports a broad set of integration protocols:

- **REST/HTTP(S)**
- **SOAP/WSDL**
- **SFTP**
- **JMS**
- **Custom adapters** implemented through Camel components
- Others

6.1.3 Messaging and Asynchronous Processing

ActiveMQ Artemis provides the messaging backbone. The ESB uses:

- Durable queues and topics
- High-availability replication
- At-least-once delivery semantics with duplicate detection
- Redelivery policies with exponential backoff
- Dead-letter queues for failed messages

This ensures reliable message processing even under node failures or downstream system outages.

6.1.4 Security Model

- The ESB enforces strict security controls:
  - **Mutual TLS** for internal service-to-service communication
  - **PKI-based** authentication for external APIs
  - **Digital signatures** when required by counterpart systems
  - **Encrypted secrets** stored in Kubernetes Secret stores

6.1.5 Error Handling and Reliability

- Each ESB service implements a standardized error-handling framework:
  - Automatic retries for transient failures
  - Dead-letter routing for persistent errors
  - Structured error events with correlation IDs
  - Circuit-breaker patterns for unstable downstream systems

This ensures predictable behaviour under failure conditions.

6.1.6 Observability and Telemetry

- All ESB services emit:
- Structured logs with correlation Id and request Id
- Transaction Log - full boundary-level message logs (headers + payloads) stored in a secure audit repository
- Health endpoints (readiness/liveness)
- Metrics (throughput, latency, queue depth, error rates)

Metrics integrate with Grafana-compatible dashboards. Logs integrate with centralized log aggregation systems.

6.1.7 Operational Tooling

The ESB supports operational visibility through:

- **Hawtio f**or inspecting Camel routes and JMS queues
- **Kaoto / Camel Caravan** for visualizing route topology
- Support for defining and deploying custom Camel routes in XML format

These tools allow operators to inspect message flows, queue states, and route execution paths.

6.2 Integration with Other Systems (CNF.73, CNF.74, CNF.75, CNF.76, CNF.77, CNF.78, CNF.80, CNF.81, CNF.82)

Apache Camel provides native support for open integration standards, enabling the solution to expose and consume REST, SOAP, HTTPS, XML/JSON, file-based, and message-driven interfaces. Event-driven communication is implemented through ActiveMQ Artemis queues and topics, while service-oriented interactions are exposed through standardized REST and SOAP endpoints. This ensures that the solution can be integrated optimally with the ESB and with all external systems.

All communication with external systems is routed through the ESB using Camel components for REST, SOAP, messaging, file exchange, and custom adapters. This guarantees consistent routing, transformation, and security policies across all integration flows. All interfaces are based on open standards such as REST/JSON, SOAP/WSDL, XML, and HTTPS. When a required interface uses a non-standard format, the integration layer adapts to it through custom Camel processors while still exposing standardized interfaces on the application side.

The integration layer supports both real-time and offline interactions. Real-time communication is handled through synchronous REST and SOAP endpoints, while asynchronous or batch-oriented processing is supported through ActiveMQ Artemis queues and topics. This dual-mode capability ensures compatibility with systems requiring immediate responses as well as those operating in event-driven or scheduled modes. Loose coupling is achieved through message-based communication, where Artemis queues decouple producers and consumers, allowing systems to operate independently of each other’s availability or performance.

Standardized interfaces are provided for accessing all key business functions, including document generation, transaction initiation, and retrieval of business-entity information. These interfaces enforce all relevant business rules and validation logic, ensuring consistent behavior across all integration channels.

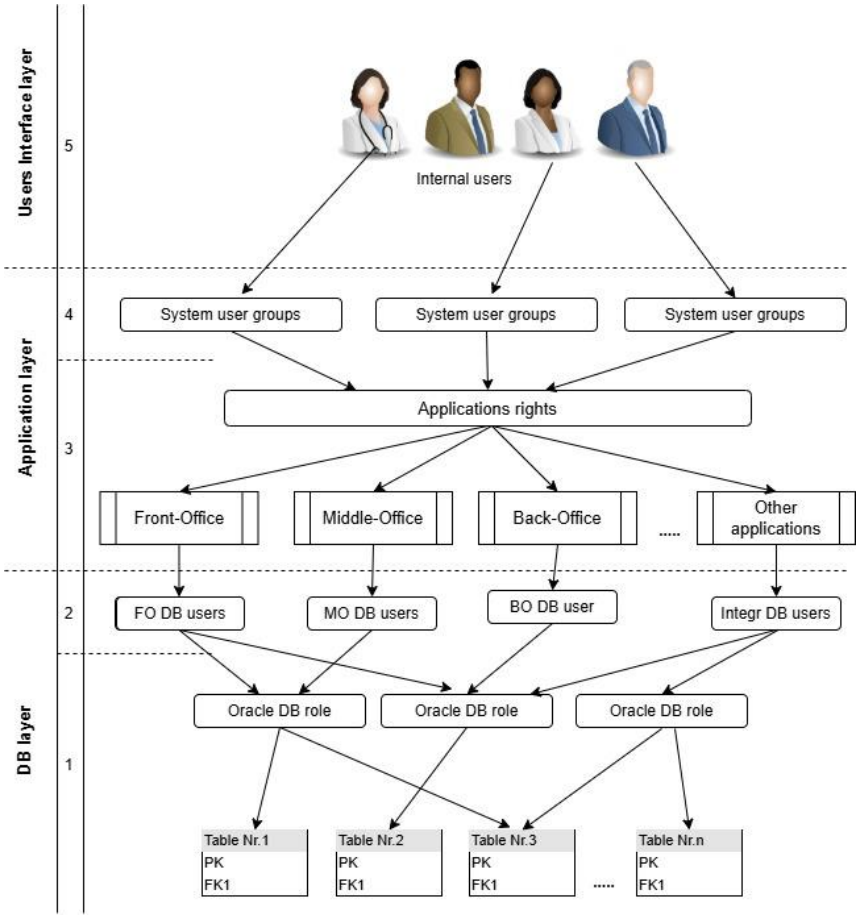
Administrators have access to comprehensive tools for managing and monitoring all external interfaces. Health endpoints, structured logs with correlation IDs, Camel route metrics, and Spring Boot Actuator data integrate with Grafana, Prometheus, and Kubernetes dashboards, providing full visibility into interface status, message flows, performance, and error conditions. All interfaces are documented using open standards, including OpenAPI/Swagger for REST services and WSDL/XSD for SOAP services, ensuring discoverability and ease of integration for external systems.

The application can generate email messages using predefined templates and send them through the configured SMTP server. This functionality is implemented through Camel's email components and can be triggered by integration events, business processes, or external requests, supporting automated communication scenarios within the broader integration landscape.

7. Security Architecture (CNF.135, CNF.136, CNF.137, CNF.139, CNF.141, CNF.148, CNF.149, CNF.150, CNF.151, CNF.184)

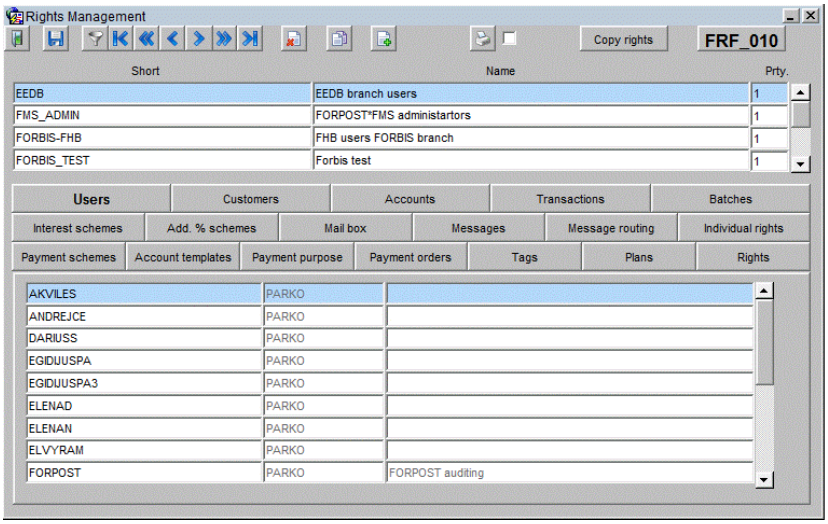
Security is implemented as a multi-layer control model covering network zoning, secure communication, identity and access management, authorization, credential protection, data validation, auditability and operational monitoring. The architecture supports Zero Trust Network Access principles by requiring components and users to communicate only through explicitly defined and secured interfaces.

7.1 Layered security zones of applications internal users (CNF.155, CNF.156, CNF.157, CNF.158, CNF.159, CNF.160, CNF.162, CNF.166)



Picture 7.1-1 "Layered security zones"

- **First level**. At the database layer, all system data is stored in database tables. Each table grants permissions to work with its data to database roles. Permissions are granted to roles not only by tables, but also by views and API program interfaces.
- **Second level**. Database users are RDBMS users who have their own accounts in the database and may connect to it using a username and password. Each user is assigned database roles, which grant permissions at the database level to work with data either directly or through the system API. All roles are secure application roles – only authorized PL/SQL package enables it. If a user connected directly to the database, bypassing the user interface, the user would not be able to perform any actions. Roles are enabled for the user only when the user connects to the system through the user interface.
- **Third level**. Application layer — application permission mechanism. System permissions are defined for working with system objects, such as clients, accounts, transactions, payments, documents, etc., as well as with system processes, such as data imports/exports, payment execution, transaction management, etc.
- **Fourth level**. System user groups. Application permissions are granted to user groups. User groups are also granted access to individual menu items of the internal system, thereby providing additional control over system users' permissions to work with system applications.
- **Fifth level**. Back-Office users performing administration functions are authenticated through the built-in Oracle DB authentication mechanism together with integrated MFA. Authentication of Internal Business users, Process management users and Reports users are performed through a dedicated third-party product — the Keycloak authentication server. After successful authentication, authorization is performed depending on the system user groups to which the user is assigned, and access to system applications is granted accordingly. The authentication mechanism used covers Front-Office and Middle-Office applications and enables easy integration of third-party applications to which the common authentication mechanism, SSO, and SLO options may be applied.



Access rights are assigned through roles and user groups. The model supports least privilege and explicit authorization for functions and data. Four-eyes controls can be implemented for agreed business workflows and approval processes. Administrative configuration changes are controlled through role restrictions, audit trails and operational procedures where NBM requires dual approval for administration, this can be addressed through workflow customization or procedure-level controls.

The application rights system allows system users to manage certain objects: customers, accounts, operation codes, account opening templates, payment orders, etc.

All objects (users, customers, operation codes, etc.) are grouped. Such grouping of objects is usually performed in separate corresponding forms. Then, each user group is granted the rights to other groups of objects, e.g. customer group, operation code group, etc.

This layered access control approach shall support NBM's alignment with the ISO/IEC 27000 family of standards by enforcing identity management, secure authentication, role-based authorization, least-privilege access, segregation of duties and traceable administrative actions. The model is aligned with relevant ISO/IEC 27001:2022 Appendix A / ISO/IEC 27002 control areas, including A.5.15 Access control, A.5.16 Identity management, A.5.17 Authentication information, A.5.18 Access rights, A.8.2 Privileged access rights, A.8.3 Information access restriction and A.8.5 Secure authentication. ISO/IEC 27001:2022 Appendix A is commonly

described as the catalogue of information security controls supporting an ISMS, while ISO/IEC 27002 provides more detailed implementation guidance for those controls.

7.2 Remote banking Authentication and Authorization

Access to Remote Banking is implemented through integration with an identity verification / identity provider service selected or approved by NBM. Alternative authentication methods may be implemented, including Multi-Factor Authentication and secure access tokens in JSON Web Token format, or other secure methods approved by NBM. The final authentication methods will be selected during the implementation project. MFA may include password, PIN, mobile signature, qualified electronic signature, cryptographic token, push approval, biometric verification, or other identity verification methods used or approved by NBM. Where JWT is used, tokens are digitally signed using a strong asymmetric signing algorithm. The preferred algorithm will be PS512 — RSASSA-PSS with SHA-512. RSA signing keys will have a modulus length of not less than 3072 bits. Token validation includes signature, issuer, audience, expiration, user identity, roles and permissions. Access to data is allowed only to authenticated and authorized users. A user is able to view, create, modify or approve only their own data or data for which they have been granted explicit rights, including rights assigned through user groups, roles or mandates. The solution prevents unauthorized access through direct object reference manipulation, identifier enumeration, replay or brute-force attempts. Where additional message-level protection is required, the solution has secure Message Authentication Code mechanism, such as HMAC-SHA-256. Certain actions may require additional user identification, confirmation or electronic signing by one or more authorized users, for example during financial transaction approval. The solution supports multi-user approval workflows, including the four-eyes principle, where required by NBM.

The described Remote Banking authentication and authorization model supports alignment with the ISO/IEC 27000 family of standards by enforcing strong identity verification, MFA-capable authentication, cryptographically protected tokens, explicit authorization and controlled access to user data and banking operations. It is aligned with relevant ISO/IEC 27001:2022 Appendix A / ISO/IEC 27002 control areas, including access control, identity management, authentication information, access rights, secure authentication, use of cryptography, and information access restriction. These controls help ensure that Remote Banking users are properly authenticated, explicitly authorized, and allowed to perform only those actions and access only those data objects for which they have approved rights.

7.3 Integration Layer Security Architecture and Identity Management (CNF.8, CNF.138, CNF.142, CNF.143, CNF.144, CNF.145, CNF.146)

All modules share a unified Java/Spring Boot codebase with enforced secure coding standards, static code analysis, dependency vulnerability scanning, and controlled library versions. This consistent approach minimizes vulnerabilities and reduces attack surface. Access credentials are never embedded in code or packaged configuration files. Instead, all sensitive values, including passwords, tokens, and certificates, are stored and managed exclusively through **Kubernetes Secrets**, which provide encrypted storage, strict access control, and secure use at runtime. All credential handling follows established security practices, ensuring confidentiality and integrity throughout the system. The application supports flexible configuration of policies governing the flow of electronic documents and ensures compliance with legal requirements for electronic signatures. It includes mechanisms for applying and verifying **advanced and qualified electronic signatures** in accordance with the European Union regulations. This guarantees the legal validity of electronically signed documents and supports secure document exchange across integrated systems.
All external interfaces are protected using secure authentication methods. Communication is performed exclusively over HTTPS with **mutual TLS**, where X.509 certificates authenticate both client and server. Depending on the integration scenario, additional authentication mechanisms - such as token-based access or signed requests - can be applied to strengthen security and ensure compliance with institutional policies.

The integration layer supports centralized authentication through **Microsoft Active Directory** using the LDAP protocol. Spring Boot and Apache Camel provide native LDAP integration capabilities, enabling the application to connect to the directory service, import user profiles, and synchronize attributes such as ID, name, surname, and email.
Mutual TLS, PKI-based authentication, encrypted secrets, and strict network policies ensure that every service verifies the identity of the caller, fully aligning with Zero Trust principles.

The described identification and access control mechanisms shall support NBM's alignment with the ISO/IEC 27000 family of standards by ensuring secure authentication, controlled and traceable access, least-privilege authorization, cryptographic protection of communications, and auditable security events.

7.4 Identity, Authentication, and Confidential Data Protection (CNF.153, CNF.154, CNF.167)

Keycloak which serves as the central authentication and authorization provider. Keycloak natively supports internationally recognized open-standard protocols such as **SAML 2.0**, **OAuth 2.0**, and **LDAP**, ensuring full interoperability with existing directory services and identity providers.
Modern multi-factor authentication (MFA) methods are supported through Keycloak's built-in capabilities, including passwords, **X.509 client certificates**, OTP/TOTP, software tokens, and other secure authentication mechanisms. When institution-specific authentication logic is required, Keycloak can be extended with custom authentication and authorization flows, ensuring that the system can adapt to future security requirements and regulatory changes.
Single Sign-On (SSO) is supported through Keycloak's native **Kerberos/SPNEGO** integration.
Confidential data protection is enforced throughout the entire architecture. Sensitive information is never stored in logs, caches, or temporary files. Logging components support configurable whitelisting and blacklisting of headers and payload fields to prevent accidental exposure of protected data. Any unauthorized attempt to access confidential information is logged with user identity, timestamp, and failure reason, ensuring full traceability. All sensitive credentials - including passwords, tokens, and certificates - are stored exclusively in **Kubernetes Secrets**, which provide encrypted storage and controlled access at runtime. Stored sensitive data is encrypted, and secure deletion techniques are applied to ensure that confidential information cannot be recovered after removal.

7.5 Input validation (CNF.108, CNF.163, CNF.169, CNF.185)

All incoming data is strictly validated and sanitized at multiple layers (UI, API, and Business Logic Layer):

- User Input (Forms, Files):
  - Validation of input formats (e.g., patterns, length, type constraints).
  - Sanitization to prevent injection attacks (e.g., SQL injection, script injection).
  - File upload validation (type, size, content checks, scan for viruses).
- External Interfaces and Integrations:
  - Schema validation (e.g., JSON/XML schema validation).
  - Strict contract-based validation for APIs.
  - Rejection of malformed or unexpected data.
- Validation Techniques:
  - Pattern matching (regular expressions).
  - Whitelisting of acceptable values.
  - Business rule validation within the Business Logic Layer.

All invalid or rejected inputs are logged for traceability and audit purposes. Logs include:

- Timestamp of the event.

- Description of the validation failure.
- Relevant details of the invalid data (where appropriate and secure).
- Logs are centrally managed and accessible for monitoring, auditing, and incident analysis.

The application includes routine procedures to detect and prevent data corruption:

- Use of hash-based checksums to verify data consistency and detect unauthorized modifications.
- Database Integrity Controls:
    - Enforcement of referential integrity through foreign keys and constraints.
    - Validation of dependencies and relationshFBS between entities.

All detected anomalies and corrective actions are fully logged.

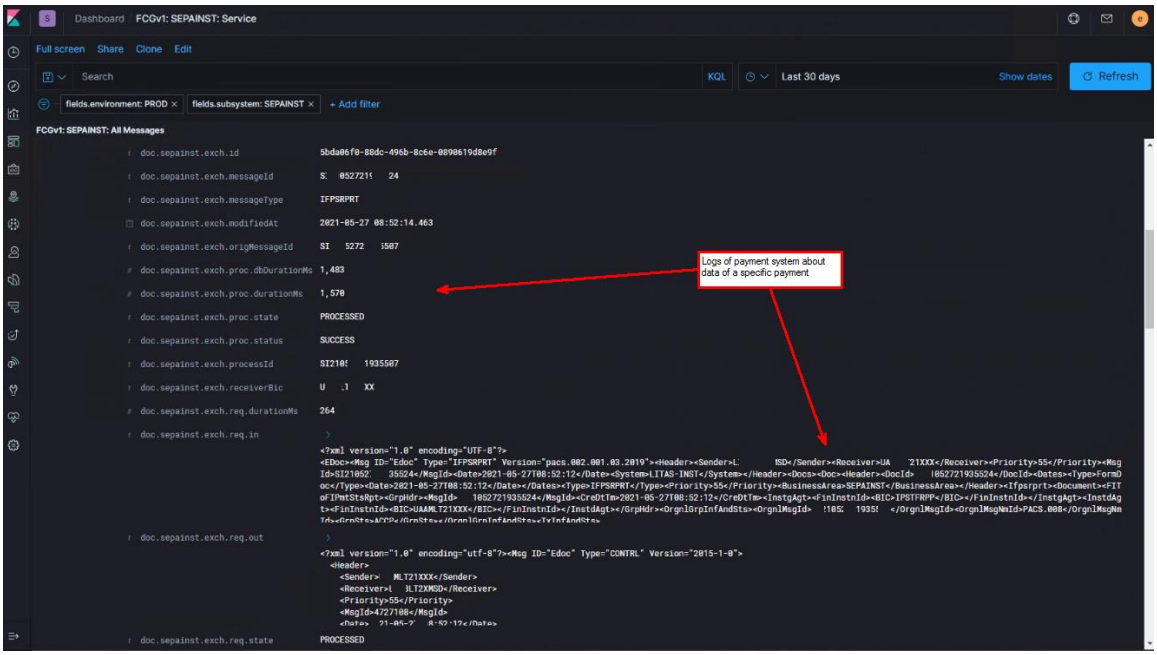
7.6 Auditing and Security Monitoring (CNF.108, CNF.170, CNF.171, CNF.172, CNF.174, CNF.175, CNF.176, CNF.182, CNF.183, CNF.185, CNF.186, CNF.194)

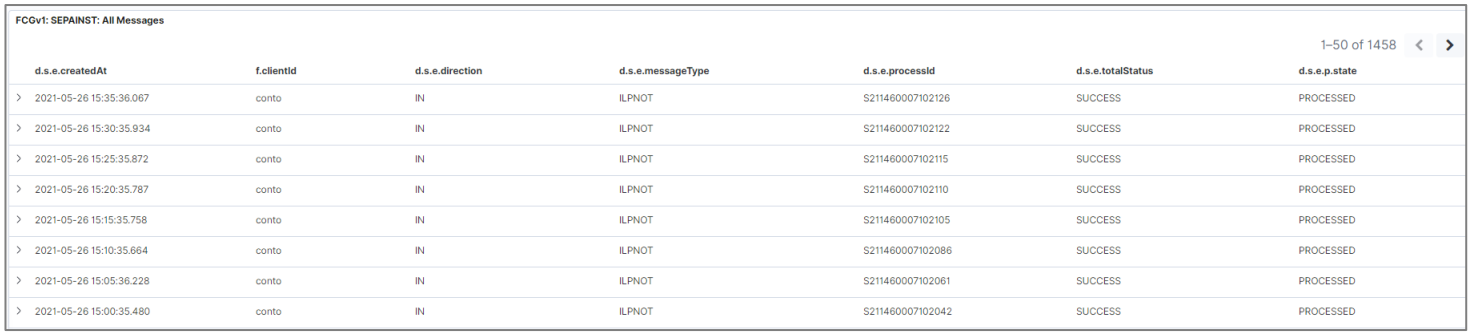
The system maintains audit trails for user-created, changed and deleted information, including the user and system time. Business logs are stored in the database, and application logs are maintained in the Java/integration layer. Correlation IDs connect logs across layers and enable reconstruction of transaction flows. For NBM, security-relevant events can be forwarded to SIEM using agreed mechanisms while avoiding exposure of passwords or confidential business data in logs.

There are two types of the logs – DB layer and applications. DB logs are based on automatically registered records using DB/tables triggers. Application logs are collected in ELK for display and analysis. There are tools for responding to critical or threshold values, both internal (within the application) and external.

The application's most relevant logs:

Logs of payment data exchange with external systems:





FCGv1: SEPAINST: All Messages

1–50 of 1458

d.s.e.createdAt	f.clientId	d.s.e.direction	d.s.e.messageType	d.s.e.processId	d.s.e.totalStatus	d.s.e.p.state
2021-05-26 15:35:36.067	conto	IN	ILPNOT	S211460007102126	SUCCESS	PROCESSED
2021-05-26 15:30:35.934	conto	IN	ILPNOT	S211460007102122	SUCCESS	PROCESSED
2021-05-26 15:25:35.872	conto	IN	ILPNOT	S211460007102115	SUCCESS	PROCESSED
2021-05-26 15:20:35.787	conto	IN	ILPNOT	S211460007102110	SUCCESS	PROCESSED
2021-05-26 15:15:35.758	conto	IN	ILPNOT	S211460007102105	SUCCESS	PROCESSED
2021-05-26 15:10:35.664	conto	IN	ILPNOT	S211460007102086	SUCCESS	PROCESSED
2021-05-26 15:05:36.228	conto	IN	ILPNOT	S211460007102061	SUCCESS	PROCESSED
2021-05-26 15:00:35.480	conto	IN	ILPNOT	S211460007102042	SUCCESS	PROCESSED

The application's universal audit mechanism for internal (DB) purposes, which keeps track of the following:

- Who (username and session id) performed an action;
- What action was performed (inserted, updated, deleted);
- On which table;
- On which columns;
- Old and new value.

The application's keeps a log dashboard for users' data audit.

- Important tables with sensitive data have journaling tables for all actions with records. These tables store the exact copy of the record at the moment of change. Thus, it is possible to see, how records have changed over time.
- Reports logs. The application keeps track of launching reports: who, when launched what report, what parameters were used. The same concerns launching interfaces, but the parameters or queries, performed on the form, are tracked selectively.
- Additional objects for auditing of web applications are:
  - User's parameters;
  - User's security means.

Various FBS subsystems have their own interfaces for recorded log events

8. Availability, Disaster Recovery and Business Continuity (CNF.10, CNF.191, CNF.192, CNF.193)

The system is a centralized hub-and-spoke system. The centre of the system is the database, which contains all business data, approximately 90% of the overall business logic, and the system parameters. All system components or component chains communicate with each other only through the database, with all operations and system parameters recorded in the database.

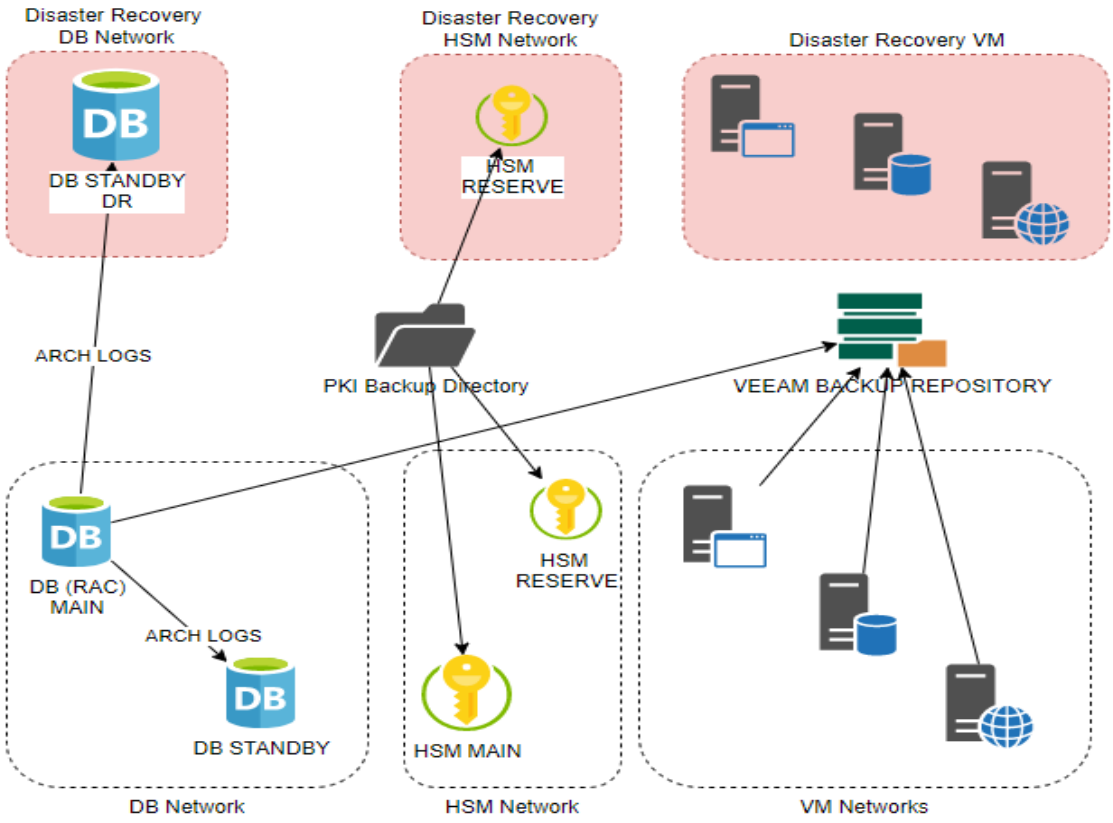
The system database ensures data integrity. Its data is continuously changed and updated.

All other system components may be rebuilt or regenerated. They change either when changes are deployed or when configuration files are modified. Peripheral system components store rarely changing data or transit data, the lifecycle of which ends once all database operations have been confirmed.
Taking the above into account, two principles shall be followed:

- **The database, as the centre of the system, must be backed up to the maximum extent possible.** Oracle provides a range of advanced tools for hot database backup and replication to different locations, such as **Oracle RMAN**, in order to minimize recovery time and avoid data loss, for example by using **Oracle Active Data Guard**.
- **Other components:**
  - Virtual machines shall be backed up once per day or immediately after changes.

- Physical devices or dedicated devices shall have a redundant deployed copy or a cold backup created once per day or after changes are deployed.

Note: Backup and recovery of VIKSVA solution nodes are performed using the above-mentioned standard Oracle and NBM infrastructure operator.



Picture 8-1. “Conceptual Backup and Recovery Scheme”

Elements of the Conceptual Backup and Recovery Scheme

Component	Description
DB Network	Subnet containing the databases.
DB (RAC)	Central system database.
DB STANDBY	Copy of the central database created using Oracle Data Guard technology.
ARCH LOGS	Transaction logs generated by the central database, which are continuously transferred to the standby database.
HSM Network	Secure subnet for HSM devices.
HSM MAIN	Primary HSM device containing PKI for applications.
HSM RESERVE	Backup HSM device containing PKI for applications.
PKI Backup Directory	Secure location for storing PKI, which can be loaded into the HSM if required.
VM Networks	Subnets containing all other VMs, which are backed up using Veeam.
Disaster Recovery	Another data center for disaster recovery purposes. The database continuously applies transaction logs using Oracle Data Guard. The HSM has preloaded PKI.

	Other VMs are either pre-restored from Veeam backup storage or kept up to date by deploying changes to System software.

9. Scalability and Performance (CNF.9, CNF.83, CNF.84)

The solution is designed to scale according to transaction volume, number of users, reporting workload and integration traffic. Stateless components can be scaled horizontally, while the database layer is scaled through Oracle sizing, RAC capabilities, storage expansion, indexing, archiving and performance tuning.

- Kubernates allows additional applications, Web/API, integration and reporting instances to be deployed and balanced.
- Integration services can be scaled per interface so high-volume integrations do not impact unrelated flows.
- Database design supports indexed access, transaction consistency, partitioning/archiving approaches and controlled query optimization.
- Caching may be used for reference or configuration data where it does not compromise banking consistency or auditability. Because core banking data must remain authoritative and auditable, caching will be applied carefully during design to avoid stale balances, duplicate transactions or inconsistent regulatory data.
- Load, performance, stress, volume and recovery tests will validate that the production sizing satisfies NBM requirements.

The following benchmark and architectural measures demonstrate that the proposed solution has been designed and previously validated for high-volume OLTP processing environments. This information will be complemented by the project-specific Performance Test Plan and Performance Test Report based on

NBM volumetry and acceptance criteria.

System performance:

- System performance depends on the volume of data stored in the OLTP database and on the underlying hardware infrastructure.
- The general performance benchmark, under medium-class server load, is based on a database containing the following volumes:

Indicator	Volume
Active customers	5,000,000
Accounts	25,000,000
Transactions per day	2,000,000
Active credit agreements	800,000
Active deposits	3,000,000
User sessions / active sessions	2,500 / 50–70
SWIFT messages per day	10,000
Payment documents, including cash settlement center and inter-branch center	200,000
Exchange transactions per day OTC transactions per month	10,000
OTC transactions per month	50,000

Under maximum load, the system response time does not exceed 5 seconds, except for report generation.

Database performance:

Database performance is ensured through the following architectural measures:

1. **Avoidance of long-running transactions**
   All transactions are designed to be short-lived in order to avoid locking issues and rollback-related performance problems.
2. **Optimized data access in accordance with Oracle recommendations**
   Data access is implemented in line with Oracle best practices. This means that all data operations are executed using optimized execution plans. Required indexes are created, and where indexed access cannot provide acceptable execution time, data denormalization is applied.
3. **Separation of data types by tablespaces**
   Different tablespaces are used for different types of data. The following tablespaces may be defined:

Tablespace	Purpose
FORPOST_ARCH	Archive tables, operational archive
FORPOST_LTA	Historical data; LTA means long-term archive
FORPOST_IDX	All indexes
FORPOST_CLOB	All LOB columns
FORPOST_TRNS	Financial transaction data
FORPOST_IMAGES	Tables containing graphical data, such as photos, scanned copies, etc.
FORPOST	General operational system data

4. **Reduced SQL parsing time**
   SQL parsing time is reduced by extensive use of bind variables and cursor parameters, minimizing the number of unique SQL statements. This reduces SGA usage and CPU time required for SQL query parsing.
5. **Applications level caching**
   Frequently used data is cached at the application level where this is appropriate for business operations.

Remote banking performance:

- Since the remote banking uses Oracle RDBMS as its data source, all database performance principles described above also apply to the portal.
- In addition, the following performance improvement principles are applied:

1. Physical separation of static resources and data.
2. Browser-side rendering of the presentation layer.
3. Browser caching of non-sensitive data, such as static texts.
4. Data caching in the web session at the Oracle RDBMS layer.
5. Data aggregation at the Oracle RDBMS layer.
6. Support for asynchronous processing.

9.1 Integration layer scalability and resource management (CNF.116, CNF.117, CNF.118)

The integration layer is designed as an elastic, cloud native platform capable of adapting to changing transaction volumes without service interruption. All components are implemented as stateless Java microservices packaged in Docker containers and orchestrated by Kubernetes, which provides the foundation for automatic horizontal scaling. This allows the system to increase or decrease processing capacity

dynamically based on real-time demand, ensuring rational and efficient use of processing resources during both peak and low-activity periods.

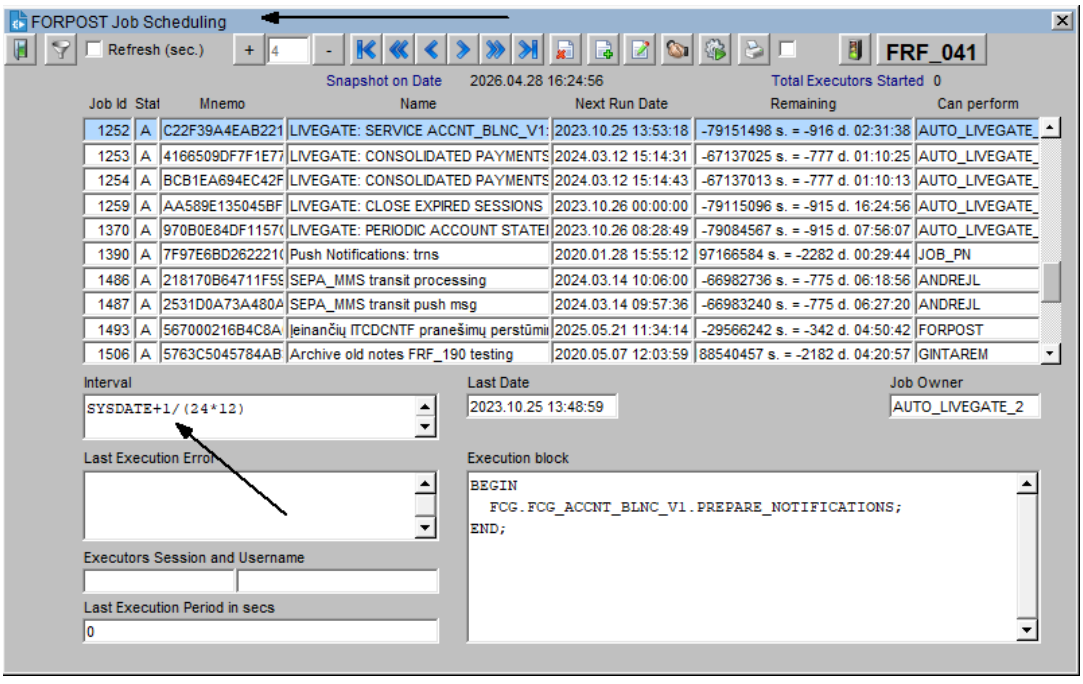
Kubernetes enables uninterrupted capacity expansion by allowing new service instances to be added while the system remains fully operational. Horizontal scaling is triggered automatically through built-in autoscaling mechanisms that monitor CPU usage, memory consumption, queue depth, request rate, or other custom metrics. As demand grows, Kubernetes provisions additional replicas of the affected microservices; when demand decreases, it scales them down to conserve resources. This ensures that the platform can respond instantly to workload changes without requiring downtime or manual intervention.

Automatic load distribution is achieved through Kubernetes service load balancing, which routes incoming traffic evenly across all active service instances. This guarantees optimal performance for latency-sensitive operations and prevents bottlenecks by ensuring that no single instance becomes overloaded.

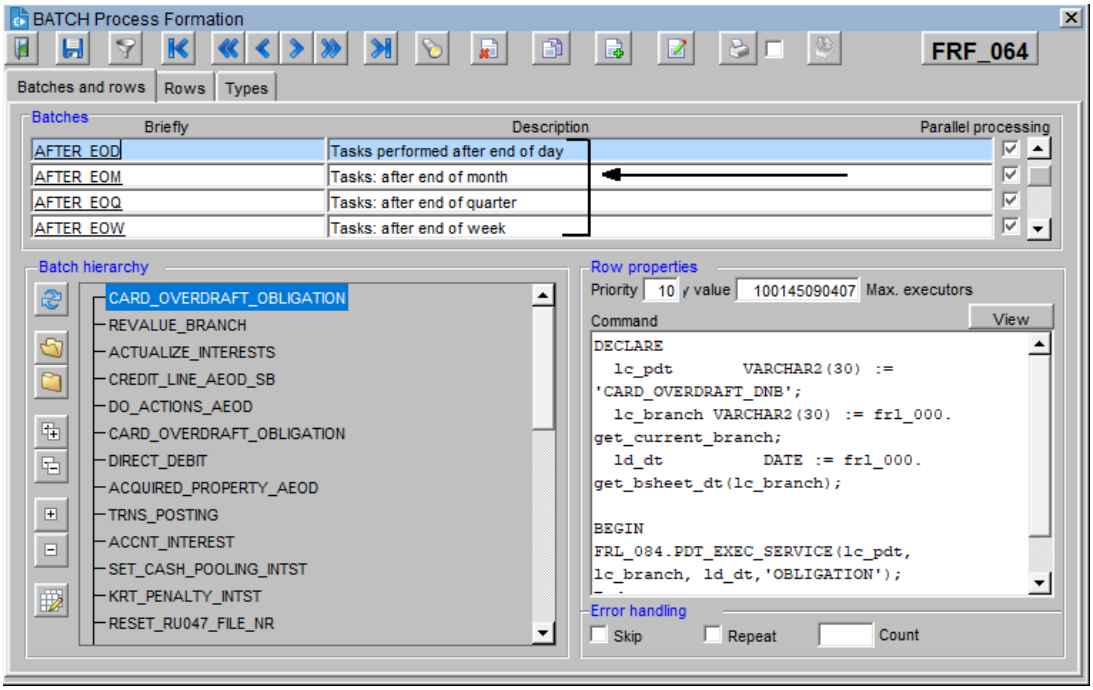
9.2 Asynchronous processing (CNF.89)

The system architecture provides asynchronous processing capabilities for resource-intensive and long-running operations, including day-end closing, scheduled background jobs during the banking day, and long-running transaction execution. These processes are executed in the background using controlled job scheduling, monitoring, and error-handling mechanisms, ensuring that normal interactive user activity remains available and responsive.

Scheduled background jobs during the banking day:



Day-end closing processes:



9.3 Database retention and archiving design (CNF.86, CNF.111, CNF.179)

The system includes a configurable **archiving, data retention and erasure subsystem**. Historical data can be managed through different lifecycle policies depending on the data category, sensitivity, and applicable business or regulatory requirements.

Archiving

In the system, archive tables are created at the database layer for important database tables, where the need for archiving is determined individually during the system development and design phase. Physically, a separate data space is allocated for these tables in the database — the **FORPOST_ARCH** operational archive. This ensures that archived data is separated from transactional data. Data archiving is performed using database triggers. Archiving for a specific table is enabled by system administrators through dedicated registry keys. Each archive table has its own dedicated registry key. The attribute indicating whether table data may be archived can also be enabled or disabled individually for each system user.

Archive tables are not accessible through user interfaces. Data deletion from archive tables is possible only through a dedicated API, which may be executed via the automatic tasks subsystem by specifying the archive table and the age of the data to be deleted.

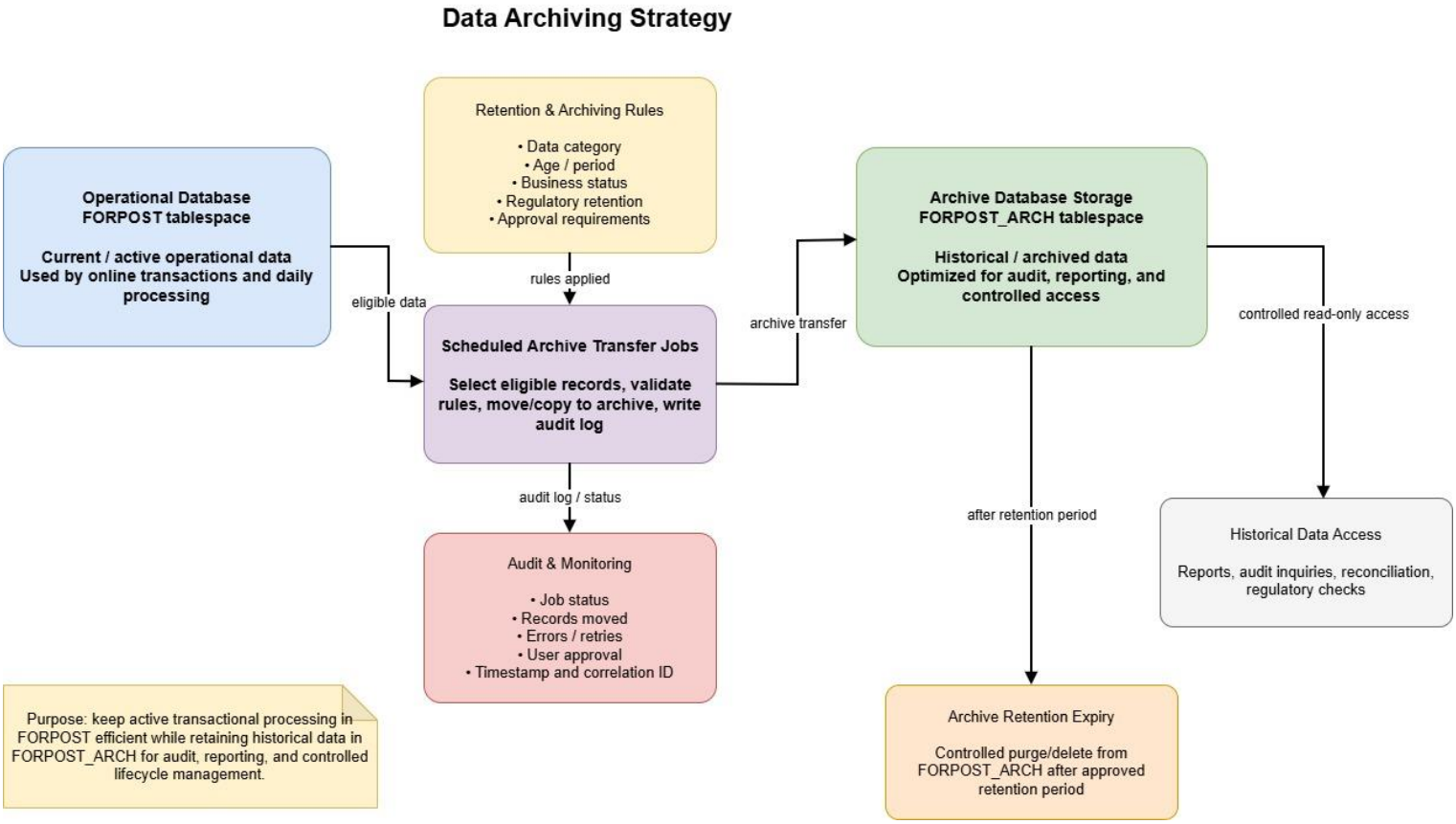
The use of common data deletion functionality ensures the protection of archived data: deletion is allowed only through the dedicated API, and the relevance of the data being deleted is controlled, preventing the deletion of archive data newer than one year.

Standard Oracle Enterprise Edition tools are used for data compression, including Basic Compression and Advanced Compression options.

Retention and erasure subsystem

For sensitive business data, such as contracts, transactions, payments, and related records, the system supports configurable data-erasure rules. Data eligible for erasure is selected according to approved rules and transferred to a temporary staging area for review. Final deletion is performed only after authorized

confirmation, ensuring controlled execution, traceability, and compliance with applicable data governance requirements.

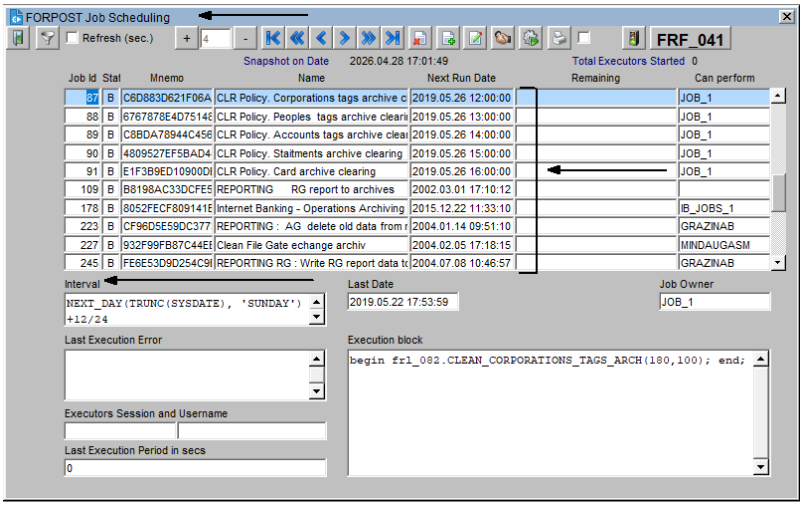


Picture 9.2-1 “Archiving strategy”

The final retention periods, archiving rules, erasure rules, approval workflow, and audit requirements will be agreed with NBM during the analysis and design phase and documented in the system administration and operation guides.

Administrator tools

Automated jobs for data transfer to archive:



10. Maintainability, Release Management and Supportability (CNF.104, CNF.104 a., CNF.104 b., CNF.104 d., CNF.105, CNF.105 a., CNF.105 b., CNF.105 c., CNF.105 d., CNF.152)

Maintainability is supported by modular architecture, documented configuration, standard technologies, controlled deployment processes and operational logging. The implementation will provide architecture documentation, installation guides, operating instructions, administrator procedures, security documentation, backup/recovery documentation, archiving procedures, interface specifications, training materials and troubleshooting guidance.

All core components of the application (front-office, back-office, business logic, and supporting modules) are delivered and maintained by a single supplier.

All application data is centralized within a single Oracle Database, ensuring consistency, integrity, and simplified data management. The solution is deployed on a standardized hardware and software environment described in "**Error! Reference source not found.**" section.

Continuous monitoring of application components (database, backend services, UI) enables early detection of anomalies. Automated alerts are triggered based on predefined thresholds (e.g., performance degradation, resource utilization).

Preventive Maintenance:

- Scheduled maintenance tasks such as data cleanup, archiving, and index optimization are implemented
- Disk space monitoring and automated alerts prevent storage-related issues
- Regular backup execution and verification ensure data recoverability

Comprehensive Monitoring:

- Real-time monitoring tools provide visibility into system health, performance, and usage across all layers:
  - Infrastructure (CPU, memory, storage)
  - Business Logic Layer (API performance, processing times)
  - Data Layer (query performance, locks, transactions)
- Centralized logging and monitoring dashboards support efficient analysis and troubleshooting

Ease of Maintenance:

- Standardized deployment and configuration processes simplify updates and fixes
- Administrative tools and scripts enable quick resolution of issues and routine operations
- Modular architecture allows isolated maintenance without impacting the entire system

10.1 Monitoring of Key Components (CNF.106, CNF.107, CNF.108)

Business Logic Layer Monitoring:

- Tracking of API response times, throughput, error rates, and service availability
- Monitoring of transaction execution times and business process flows

Data Layer Monitoring:

- Continuous observation of query performance and slow-running SQL
- Monitoring of locks, deadlocks, and transaction contention
- Tracking of database sessions, connections, and resource usage

Operational Load Tracking:

- Real-time monitoring of CPU, memory, disk I/O, and network utilization

The system includes automated self-diagnostic mechanisms to continuously assess component health:

- Built-in health checks for services, database connectivity, and critical processes
- Automatic detection of failures, degraded performance, and unavailable components
- Correlation of logs and events to support root cause analysis

Alerts are triggered based on configurable thresholds and anomaly detection rules.

Notifications include: clear description of the issue, affected components, severity level and potential impact.

Alerts are delivered via: email notifications, monitoring dashboards and integration with incident management tools.

10.2 Technical Documentation (CNF.109)

Technical Architecture Documentation includes:

- Detailed description of the overall system architecture, including: Presentation Layer (React front-office, Oracle Forms back-office), Business Logic Layer (services, APIs, PL/SQL components) and Data Layer (Oracle database structures and interactions)
- Architecture diagrams illustrating component interactions, data flows, and integration points
- Description of deployment topology and infrastructure setup

Installation Guides includes:

- Step-by-step instructions for installing all components
- Environment prerequisites (hardware, OS, software dependencies)
- Configuration of environments (development, testing, production)

Configuration and Operational Maintenance includes:

- Detailed configuration guidelines for all components:
- Operational procedures including:
    - Startup and shutdown processes
    - Backup and recovery procedures
    - Monitoring and logging configuration
    - Routine maintenance tasks (e.g., cleanup, archiving, performance tuning)

Developer Guides includes:

- Instructions for extending or customizing functionality for internal development by NBM
- Examples and templates to support development and integration activities

Integration and Extension Guidance includes:

- Clear instructions on how NBM-developed components can be:
  - Integrated into the existing architecture
  - Deployed and configured within the system
  - Maintained without impacting core components
- Definition of supported extension points and interfaces to ensure safe customization

10.3 Flexibility (CNF.92)

Users can personalize layouts according to their preferences. The solution allows the development of new forms and screens to interact with existing business logic. New forms can consume exposed APIs or PL/SQL procedures without modifying core logic. Reusable components and templates accelerate development and ensure consistency. All customizations and new forms operate within defined architectural boundaries. Access to business logic is governed through secure and well-defined interfaces.

10.4 Portability (CNF.114)

Portability is achieved by applying the following architectural principles and internal procedures

Portability area	Implementation Approach
Consistent deployment model	The solution can be deployed in production, test, UAT, and development using the same architecture and deployment approach.
Externalized configuration	Environment specific settings are separated from applications code: URLs, ports, database connections, certificates, credentials, integration endpoints, queues, scheduled jobs.
Documented dependencies	All required software, middleware, database, operating systems, libraries, versions, network ports, certificates, and third-party components are documented. The proposed infrastructure stack is based on enterprise-supported Oracle and Red Hat products where applicable, which supports predictable dependency management, lifecycle control, security patching, and repeatable deployment across environments.
Database and data portability	Database schema, migration scripts, reference data, and controlled data transfer between environments are supported. Production data used in non-production environments is masked or anonymized where required.
Repeatable deployment and migration process	Deployment and migration between environments are performed using step-by-step documented procedures, scripts, packages and Jenkins-assisted CI/CD mechanisms.
Validation and rollback	After upgrades, the system is checked using smoke tests, integration tests, configuration checks, and operational validation. Rollback procedures are documented.

10.5 Upgrade governance and environments (CNF.112)

The conceptual scheme for upgrade deployment and system environments is presented below:

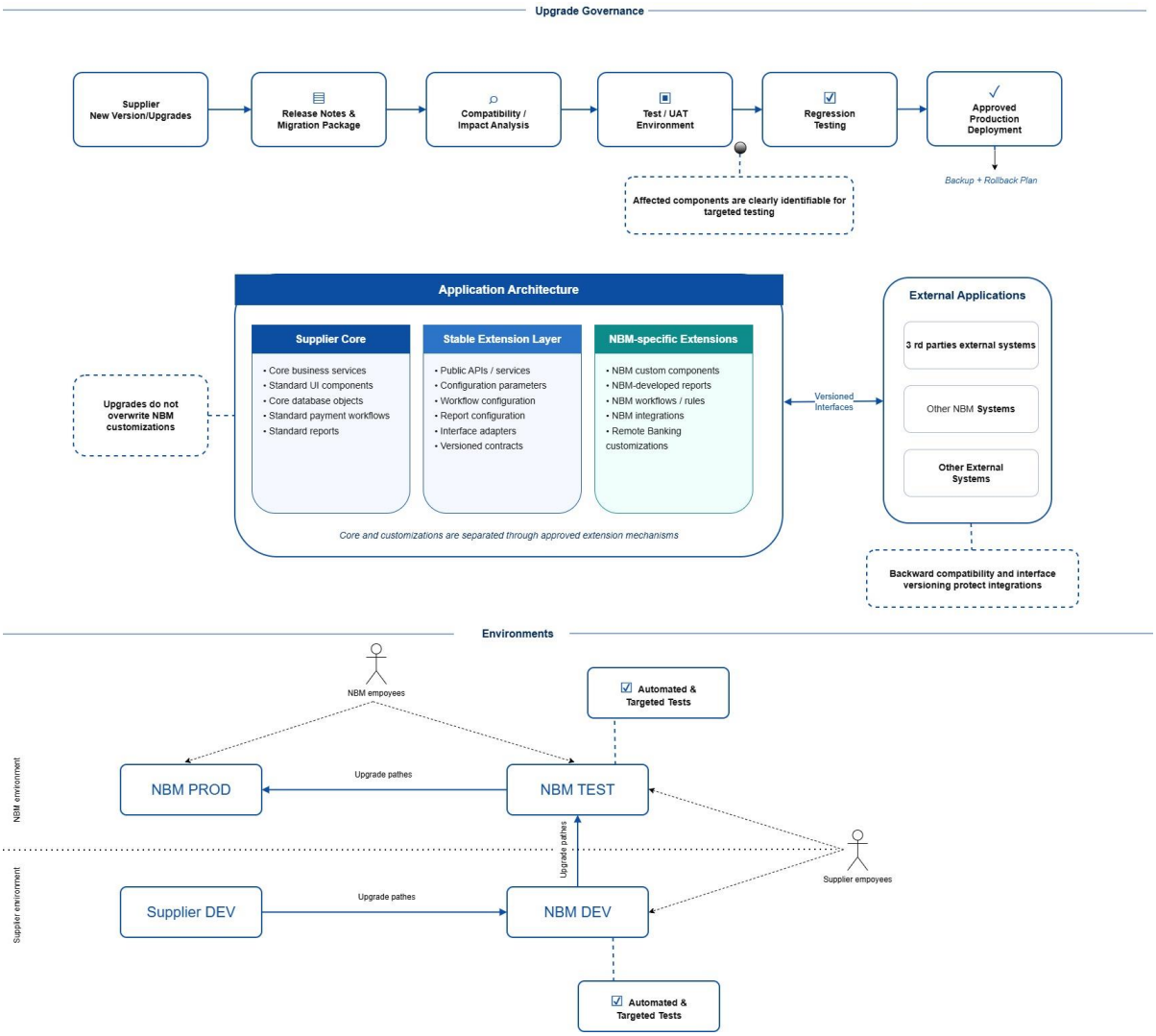


Figure 13. 10-1 “Upgrade governance and environments”

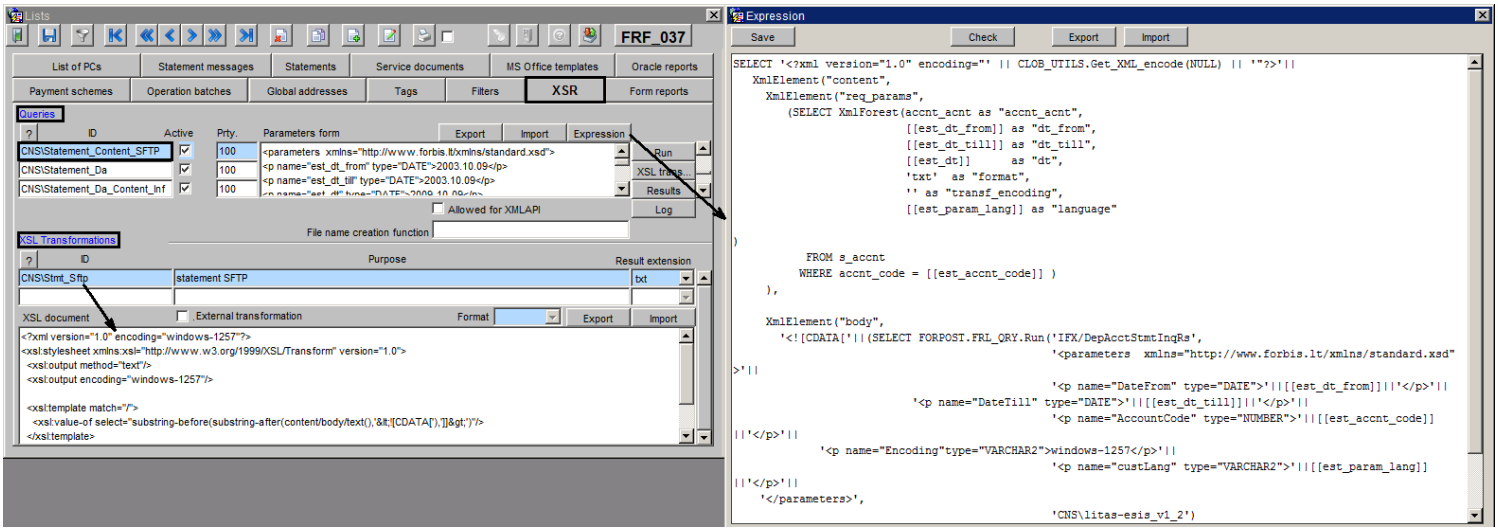
10.6 Development Standardization (CNF.104 c., CNF.115)

Each module follows the same delivery rules, including identical build pipelines, dependency structures, and coding standards. All services are packaged as Docker images, and a single image per module is reused consistently across all delivery stages—development, testing, staging, and production. This approach eliminates environment-specific discrepancies, reduces integration risks, and ensures predictable behavior throughout the entire lifecycle of the application. The uniform toolchain and consistent CI/CD processes significantly simplify maintenance, testing, and long-term evolution of the system.

To support operational reliability and rapid incident resolution, the application includes mechanisms for automated notification of critical errors. All integration services expose structured metrics, health information, and alert signals through Spring Boot Actuator and Apache Camel health endpoints. These are collected centrally and monitored through Grafana, which evaluates error codes, timestamps, correlation identifiers, and other diagnostic details. Grafana alerting rules can be configured to automatically notify responsible parties—including the software manufacturer—via email, webhook, or other supported channels whenever a critical error occurs. This ensures that severe issues are detected immediately and accompanied by sufficient diagnostic information to enable rapid identification and resolution of root causes.

11. Reporting and Data Access for Users (CNF.93, CNF.94, CNF.95)

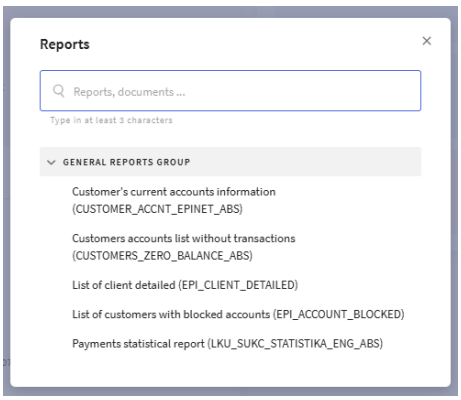
Reporting is supported through XSR/Jasper reports services and MS 365 report services. The system allows creating reports through configuration by defining query logic and output formats. Reports tool supports dynamic queries, selection criteria to refine report content and customizable layouts, columns, grouping, and ordering. Example of XSR/Jasper reports:

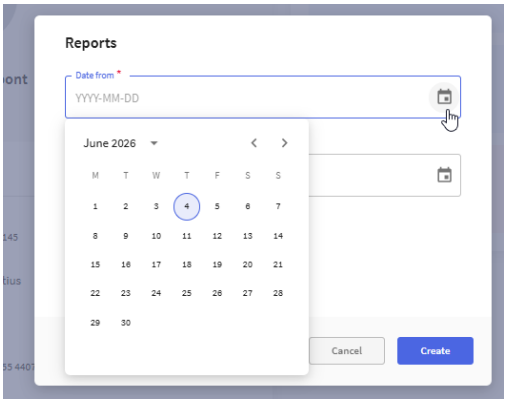


The implementation scope will include analysis and setup of the required customized reports. Reporting access is controlled by user roles and the data dictionary will support users involved in defining and maintaining reports.

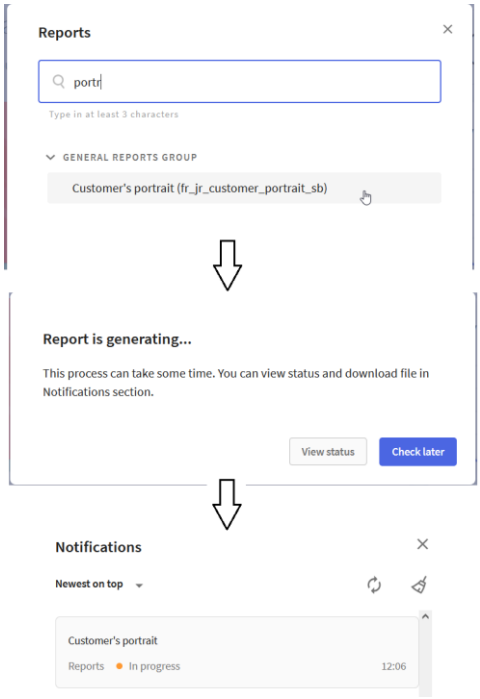
- JasperReports services provide PDF, HTML, CSV, JSON, XML reports generation.
- XSL engine provide XML, HTML, CSV, TXT, DBF reports generation.
- MS 365 report services support Word and Excel output where required.
- Reports are based on controlled database queries and role-based data access.
- Performance considerations, indexes and reporting data access patterns will be reviewed during report design.

Reports can be generated automatically in response to specific business events (e.g., transaction completion, status changes) or scheduled (e.g., daily, weekly, monthly) using background jobs. Generated reports can be stored within the application repository for later access or automatically distributed (e.g. via email) to predefined users or groups. Users can define and manage report generation rules through intuitive interfaces. Access control ensures only authorized users can configure or receive reports. Bank employees can generate reports from the list according to their access rights.





11.1 Parameterization of reports generation (CNF.85)

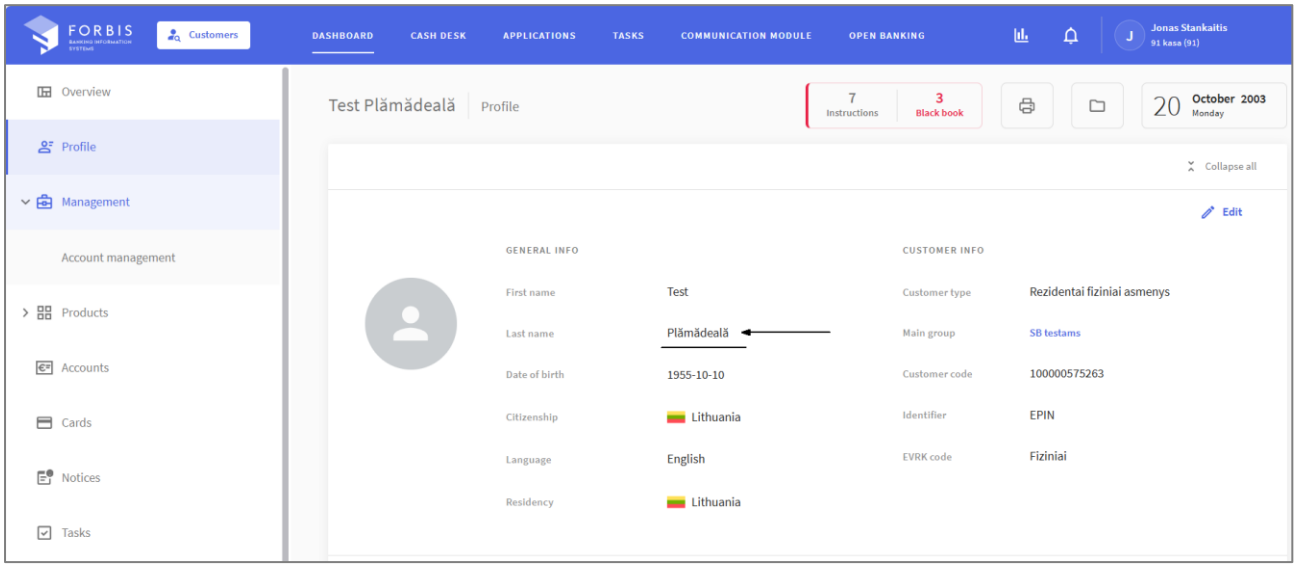


The application allows creating new or customizing the existing reports. Back-office administrators can create/modify reports, combine them into groups, change report parameters. If report generation is expected to take a long time, check the "Long execution" checkbox. In this case, report generation will be performed asynchronously, and the user will be able to monitor the progress in a separate window:

11.2 Unicode format (CNF.45)

The applications support the creation, modification, processing, storage, and retrieval of textual data in Unicode format. Users can enter, edit, save, and access text containing Unicode characters, including special symbols and multilingual characters, through

the applications user interface. The system preserves Unicode text during processing and storage, ensuring that the entered data is displayed correctly when retrieved or viewed later. The Unicode text entry and display is provided in the screenshot below:



12. Data Migration and Implementation Readiness (CNF.29)

The supplier will provide a complete migration methodology and detailed migration plans. The approach includes source data assessment, mapping, cleansing, transformation, ETL tooling, import procedures, reconciliation, exception handling, rollback/recovery planning and go-live migration rehearsal. The data model is designed to fully support migration from existing systems, ensuring complete and accurate data transfer in line with NBM requirements.

- The schema aligns with source system structures and data semantics, enabling consistent mapping of entities and attributes.
- Data types, formats, and value ranges are defined to match or accommodate existing system standards.

- Migration processes include validation, transformation, and reconciliation controls to ensure data completeness and correctness.

12.1 Data Migration Approach and Methodology

Overview and Core Migration Tool

Our approach to data migration is centred around our robust, highly secure XMLAPI interface, which serves as the primary software tool and gateway for all data ingestion. Rather than relying on direct, unvalidated database inserts, our methodology ensures that all historical and operational data from NBM's legacy applications is transformed into standardized XML formats and imported via the XMLAPI. This approach guarantees that imported data undergoes the exact same rigorous business logic, transaction control, and context management (including NLS parameters) as data entered manually into the live system. We will provide NBM with comprehensive documentation of the required XML structures, which will serve as the exact blueprint for data extraction and transformation.

Planning, Mapping, and Methodology

The migration process begins with the development of a Detailed Data Migration Plan, outlining all timelines, milestones, environments, and responsibilities. During the initial phase, we will perform comprehensive Data Mapping. Using the data structure information provided by NBM, we will map the source fields from NBM's legacy applications directly to our target XML schemas. This mapping exercise will define the transformation rules required for external systems or middleware to generate the correct XML files for import.

Quality Rules, Validation, and Cleansing

Data quality is paramount to our migration strategy. Quality rules are inherently established and strictly enforced by the XMLAPI. Every implementation of the XMLAPI incorporates specific, predefined business validation rules.

To facilitate Data Cleansing and Enrichment, we will conduct iterative trial migrations in isolated test environments. By running preliminary extracts through the XMLAPI, we will proactively identify data quality gaps, structural anomalies, and missing elements. The validation errors returned during these trial runs will highlight precisely where NBM's source data requires cleansing or enrichment prior to the final production cutover.

Data Import Execution and Exception Handling

Once the data sets are prepared, the actual Data Import is seamlessly executed by feeding the generated XML files into the XMLAPI. Before any data is permanently committed to the new solution, the XMLAPI automatically acts as a validation gateway, verifying and validating the structural integrity and business quality of the incoming data sets.

During the import process, our system utilizes a comprehensive, common error handling framework. Any records that fail business validation or structural checks generate specific, detailed error logs. This allows for the immediate identification, isolation, and resolution of exceptions and errors without halting the entire migration batch.

Transaction Control and Reconciliation

To guarantee zero data loss and absolute data integrity, the XMLAPI natively handles strict transaction controls. Furthermore, we will establish clear Data Reconciliation Criteria in collaboration with NBM. Post-import, we will compare the volume and financial totals of the source NBM data against the successfully

processed XMLAPI transactions. This multi-layered approach—combining structural validation, business rule enforcement, detailed error logging, and post-import reconciliation—ensures a secure, transparent, and fully audited transition to the new solution.

12.2 Data Extraction from Current Systems

The data migration methodology includes a controlled approach for extracting data from current NBM systems. The extraction method will be selected according to the technical capabilities of each source system, the type of data to be migrated, data volumes, data quality, and the agreed migration schedule.

Data may be extracted using one or more of the following methods:

Extraction method	Description
Database queries / views	Data is extracted directly from source databases using agreed SQL queries, database views, or controlled extraction procedures.
Export scripts	Repeatable scripts are used to export data from legacy systems in agreed formats.
File-based export	Data is exported into structured files such as CSV, XML, JSON, Excel, TXT, or other agreed formats.
API-based extraction	Where supported by the source system, data may be extracted through application APIs or services.

12.3 Data Structure Mapping Approach

The data migration methodology includes a structured **source-to-target mapping approach** for all data structures included in the migration scope. The purpose of this activity is to define how data from the current NBM systems will be transformed, validated, and loaded into the target System.

For each migration data domain, analysis will be performed:

Mapping area	Description
Source data structures	Data is extracted directly from source databases, database views, or controlled extraction procedures.
Target data structures	Repeatable scripts are used to export data from legacy systems in agreed formats.
Reference data/classifiers	Data is exported into structured files such as CSV, XML, JSON, Excel, TXT, or other agreed formats.
Transformation rules	Where supported by the source system, data may be extracted through application APIs or services.
Data quality rules	Handling of missing values, duplicates, invalid formats, inconsistent references, and obsolete records.
Historical data rules	Mapping of historical records, archive data, closed accounts/contracts, and old transaction references where applicable.
Validation and reconciliation rules	Record counts, control totals, balances, mandatory-field checks, and referential-integrity checks.

12.4 Data Reconciliation Approach

The data migration methodology includes a structured reconciliation approach to confirm that migrated data is complete, accurate, consistent, and usable in the target System. Reconciliation will be performed after each trial migration and after the final production migration.

Reconciliation area	Description
Record-count reconciliation	Comparison of the number of records extracted from source systems, loaded into the staging area, and imported into the target System.
Control totals	Comparison of key numeric totals, such as balances, transaction amounts, quantities, accounting totals, securities positions, or other agreed control values.
Mandatory-field validation	Confirmation that all required target System attributes are populated according to migration rules.
Referential-integrity checks	Validation that migrated records maintain correct relationshFBS, for example customer–account, account–transaction, contract–payment, security–position relationshFBS.
Duplicate checks	Verification that migration did not create duplicate records where uniqueness is required.
Source-to-target identifier matching	Use of cross-reference tables to confirm traceability between legacy identifiers and new target System identifiers.
Business-report comparison	Comparison of selected reports generated from source and target systems to validate business consistency.
Exception analysis	Review of rejected records, warnings, differences, and unresolved migration issues.

Each reconciliation run will produce a **Migration Reconciliation Report**. The report will include the migration run identifier, data domain, source totals, target totals, differences, rejected records, identified issues, correction actions, responsible owners, and approval status.

Final migrated data will be accepted only after successful reconciliation and business-owner sign-off according to the agreed migration acceptance criteria.

12.5 Migration Recovery Plan

The data migration methodology includes a recovery plan for each key phase of the migration process. The purpose of the recovery plan is to ensure that, in case of migration failure, the migration process can be resumed from the last feasible and validated phase without unnecessary repetition of already completed activities and without disrupting NBM business processes.

Migration phase	Recovery approach	Responsibility
Data extraction	If extraction fails, the extraction can be restarted from the affected source system or data domain using repeatable extraction scripts and extraction logs.	NBM
Staging	If staging fails, the affected data package can be reloaded into the staging area from the original extracted files or source export.	NBM
Transformation and mapping	If transformation fails, the affected transformation rules can be corrected and re-executed from the last valid staging dataset.	Supplier
Data import	If import fails, rejected records are isolated, errors are corrected, and the affected import batch is reprocessed according to the import dependency sequence.	NBM, Supplier
Validation and reconciliation	If reconciliation differences are identified, they are analyzed and corrected through agreed data correction, remapping, or re-import procedures.	NBM, Supplier

Migration phase	Recovery approach	Responsibility
Final cutover	If final migration or go-live cannot be completed successfully, rollback/fallback procedures are applied according to agreed go/no-go criteria.	NBM, Supplier

12.6 Go-live Plan for Data Migration and Production Launch

The data migration methodology includes a detailed Go-live Plan to ensure that all required data is available, validated, reconciled, and approved at the time the solution is launched in production.

The Go-live Plan will define the complete set of activities required for production launch using the following structure:

Go-live element	Description
What	Activities to be performed, including final extraction, migration freeze, staging, transformation, final import, validation, reconciliation, access verification, integration checks, backup, and post-go-live monitoring.
Who	Responsible roles from the supplier and NBM, including migration team, technical team, business owners, system administrators, security team, database administrators, integration team, and project management.
When	Detailed timeline for pre- go live, final migration window, production cutover, go/no-go decision, go-live execution, and post-go-live support.
How	Execution method, including scripts, import jobs, migration procedures, validation checks, reconciliation controls, approval workflow, escalation process, and rollback/fallback procedures.
Where	Target environments and locations where activities are performed, including staging environment, production environment, source systems, integration components, database layer, and operational support channels.

The Go-live Plan will include:

- final migration schedule and cutover timeline;
- data freeze and delta-data handling rules;
- final extraction and import procedures;
- production backup and restore points;
- validation and reconciliation checkpoints;
- go/no-go decision criteria;
- responsible persons and escalation path;
- communication plan;
- rollback and fallback procedures;
- post-go-live monitoring and support activities.

Before production go-live, the plan will be tested through trial migrations and cutover rehearsals. Final go-live approval will be based on successful migration execution, reconciliation results, production readiness checks, integration availability, and business-owner sign-off.

12.7 ETL Toolset and Migration Utilities

The data migration process will be supported by a specialized ETL toolset and migration utilities provided by the Supplier. The approach does not require a separate commercial ETL platform. The ETL functions will be

implemented through controlled migration scripts, database procedures, staging tables, transformation rules, validation jobs, import/export utilities, error-handling mechanisms, migration logs, and reconciliation reports.

12.8 Completeness and Data Integrity

To ensure that all selected data is migrated fully and accurately without compromising relationshFBS or integrity, the following measures are applied:

- **Schema Alignment and Mapping**
  - The target data model is designed to align with source system structures and semantics.
  - A detailed field-by-field mapping specification is developed, defining:
    - Attribute correspondence
    - Data transformations (if required)
    - Referential relationshFBS
  - Referential integrity (parent-child relationshFBS, dependencies) is preserved through controlled import sequencing.
- **XMLAPI-Based Controlled Data Ingestion**
  - All data is converted into standardized XML structures and imported via XMLAPI.
  - This ensures:
    - Uniform enforcement of business rules
    - Validation of mandatory fields, formats, and constraints
    - Preservation of logical relationshFBS
- **End-to-End Validation and Reconciliation**
  - Multi-level validation:
    - Structural validation (schema compliance)
    - Business rule validation
  - Post-migration reconciliation:
    - Record counts
    - Financial totals
    - Control balances
  - Full auditability of migrated data sets
- **Exception Handling**
  - Errors are logged at record level with detailed diagnostics.
  - Failed records are isolated and corrected without impacting overall migration batches.

12.9 Timeliness (Minimal Business Disruption)

To ensure fast migration with minimal interruption to business operations, the approach includes:

- **Phased and Iterative Migration Strategy**
  - Trial migrations executed in parallel environments
  - Early identification of data issues reduces production delays
- **Automation via XMLAPI**
  - Automated ingestion eliminates manual processing
  - Parallel batch processing supported
- **Cutover Optimization**
  - Final migration ("production cutover") is streamlined:
    - Pre-validated datasets
    - Reduced downtime window
  - Option to implement incremental/delta migration for near-zero disruption
- **Clear Migration Plan**

        - Includes timelines, milestones, rollback strategies

12.10 Efficiency and Cost Optimization

To ensure cost-effective migration aligned with business value, the methodology emphasizes:

- **Reuse of Standard Tools (XMLAPI)**
    - Eliminates need for custom data-loading tools
    - Reduce development and maintenance costs
- **Early Data Quality Detection**
    - Trial runs identify issues early → avoids costly late-stage corrections
- **Clear Scope Definition**
    - Migration scope (data categories and structure) agreed during Analysis & Design phase
    - Prevents unnecessary migration of low-value or obsolete data
- **Automation and Repeatability**
    - Reduces manual effort and error correction cycles
    - Enables reuse of scripts/processes for future needs

12.11 Security and Data Protection

To ensure no compromise to data confidentiality, integrity, or access control, the following measures are implemented:

- **Secure Data Handling**
    - Data transferred through controlled pipelines
    - No direct database inserts or uncontrolled access points
- **XMLAPI as Controlled Gateway**
    - Acts as a secure ingestion layer enforcing:
        - Authentication and authorization
        - Data validation before acceptance
- **Environment Segregation**
    - Separate environments:
        - Development
        - Test
        - Production
    - Sensitive data handled according to security policies
- **Audit and Logging**
    - Full traceability of:
        - Data imports
        - Errors
        - Corrections
    - Supports compliance and audit requirements
- **Data Leakage Prevention**
    - Controlled access to migration datasets
    - Use of masked/anonymized data where applicable during testing phases

12.12 Role of the Supplier

The Supplier plays a central and critical role in ensuring migration success:

- **Provision of Tools and Framework**

    - ETL processing framework
    - XMLAPI and integration specifications
    - XML schema definitions and documentation
- **Methodology Ownership**
    - Definition of migration plan, strategy, and execution model
    - Governance of migration lifecycle
- **Technical Expertise**
    - Proven experience with:
        - Large-scale data migrations
        - Schema mapping and transformation
        - Financial and transactional data reconciliation
- **Collaboration with NBM**
    - Joint definition of:
        - Data sets to migrate
        - Validation rules
        - Reconciliation criteria
    - Support in data cleansing and preparation

12.13 Data Structure and Governance (Analysis & Design Phase)

In line with NBM requirements:

- **Joint Agreement on Data Scope**
    - Data categories and structures defined collaboratively
    - Aligned with business objectives
- **Preservation of Existing Formats**
    - Data retained in current format wherever feasible:
        - Value ranges
        - Field lengths
        - Data types
- **Controlled Exceptions Process**
    - Any deviations:
        - Proposed by the Supplier
        - Justified with technical/business reasoning
        - Subject to formal NBM approval

12.14 Migration Lifecycle Summary

- **Planning:** Define scope, timelines, roles
- **Data Mapping:**
    - Source-to-target mapping
    - Transformation rules
- **Trial Migration Cycles**
    - Identify data quality issues
    - Refine mappings
- **Data Cleansing:** Resolve inconsistencies in source data
- **Final Migration Execution:** XML-based import via XMLAPI
- **Validation & Reconciliation:** Ensure completeness and correctness
- **Go-Live Support:** Monitor and resolve post-migration issues

13. Usability and User Experience

The system provides role-based user interfaces with consistent navigation and clear separation between internal and external user groups. Graphical interfaces expose authorized business functions, provide validation feedback, support context help/tooltFBS and enable efficient daily banking work. Romanian and English localization will be implemented and verified with NBM through a controlled terminology glossary and user acceptance testing.

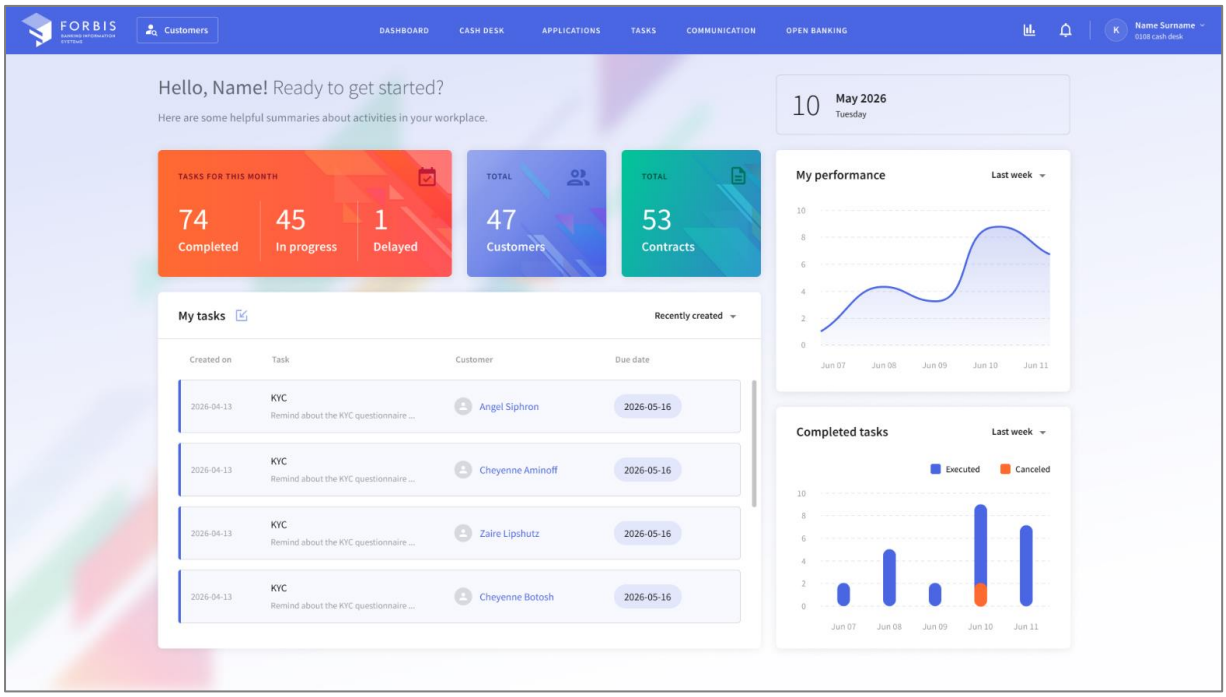
- Functions are grouped according to user role and operational context.
- Users receive clear feedback on validation errors, status changes and required actions.
- Role-based dashboards and notifications can be configured for approvals, exceptions and assigned tasks.
- Documentation and training materials will support user onboarding and operational adoption.

13.1 Intuitive interface (CNF.119, CNF.120, CNF.121, CNF.122, CNF.124, CNF.125, CNF.129, CNF.131, CNF.134, CNF.187)

The application is designed around a unified graphical interface that provides users with centralized access to all functions authorized for their role. The interface architecture ensures consistent navigation patterns, predictable interaction behaviour, and efficient task execution across the entire system.

The solution follows a role-based information architecture approach, allowing business users, analysts, and administrative personnel to access relevant functionality through a coherent and streamlined user experience. Navigation elements, page layouts, action placement, filtering mechanisms, and interaction models are standardized throughout the application to minimize user confusion and reduce operational complexity.

Special attention is given to high-frequency workflows and operational efficiency, ensuring that users can complete critical tasks with a minimal number of interactions while maintaining full visibility of relevant contextual information.



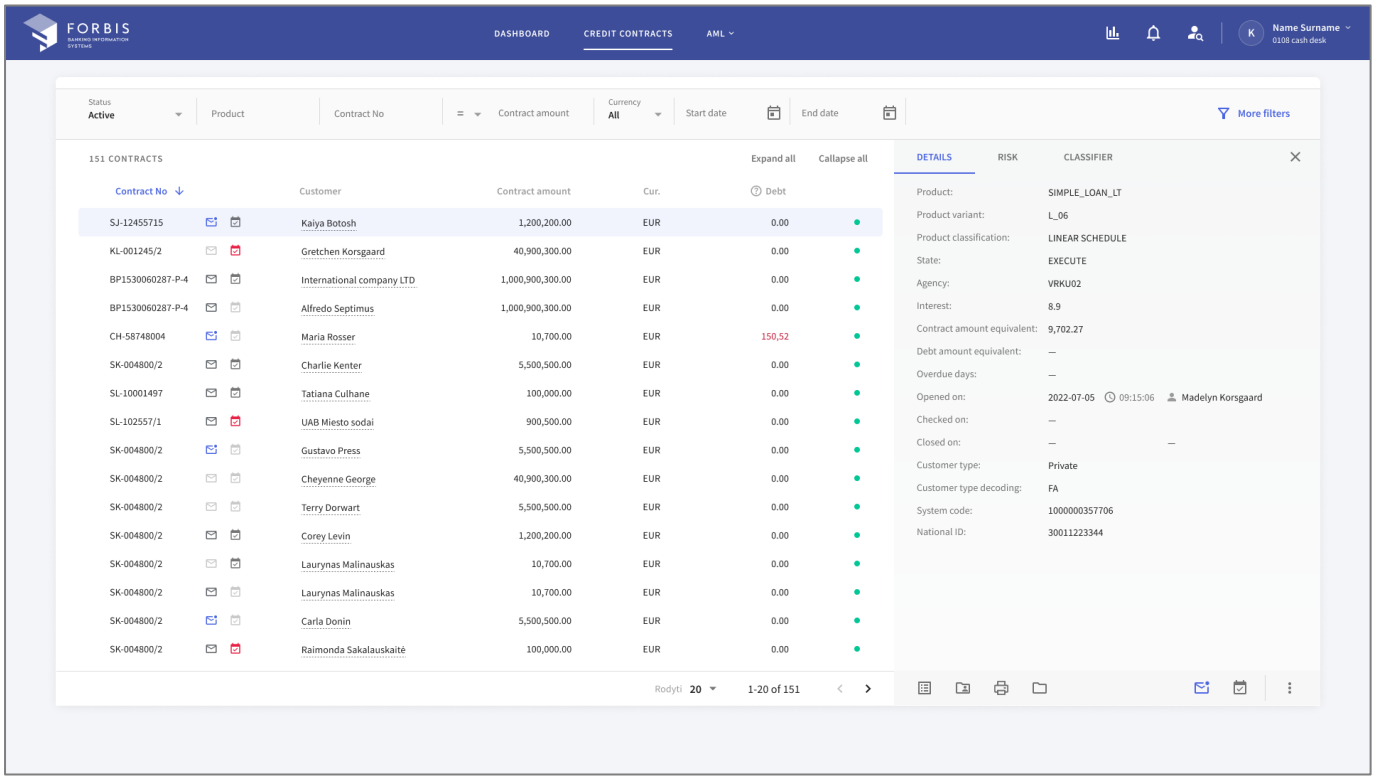
The user interface is designed according to established usability and user-centred design principles, with particular focus on clarity, readability, cognitive ergonomics, and operational efficiency in enterprise environments.

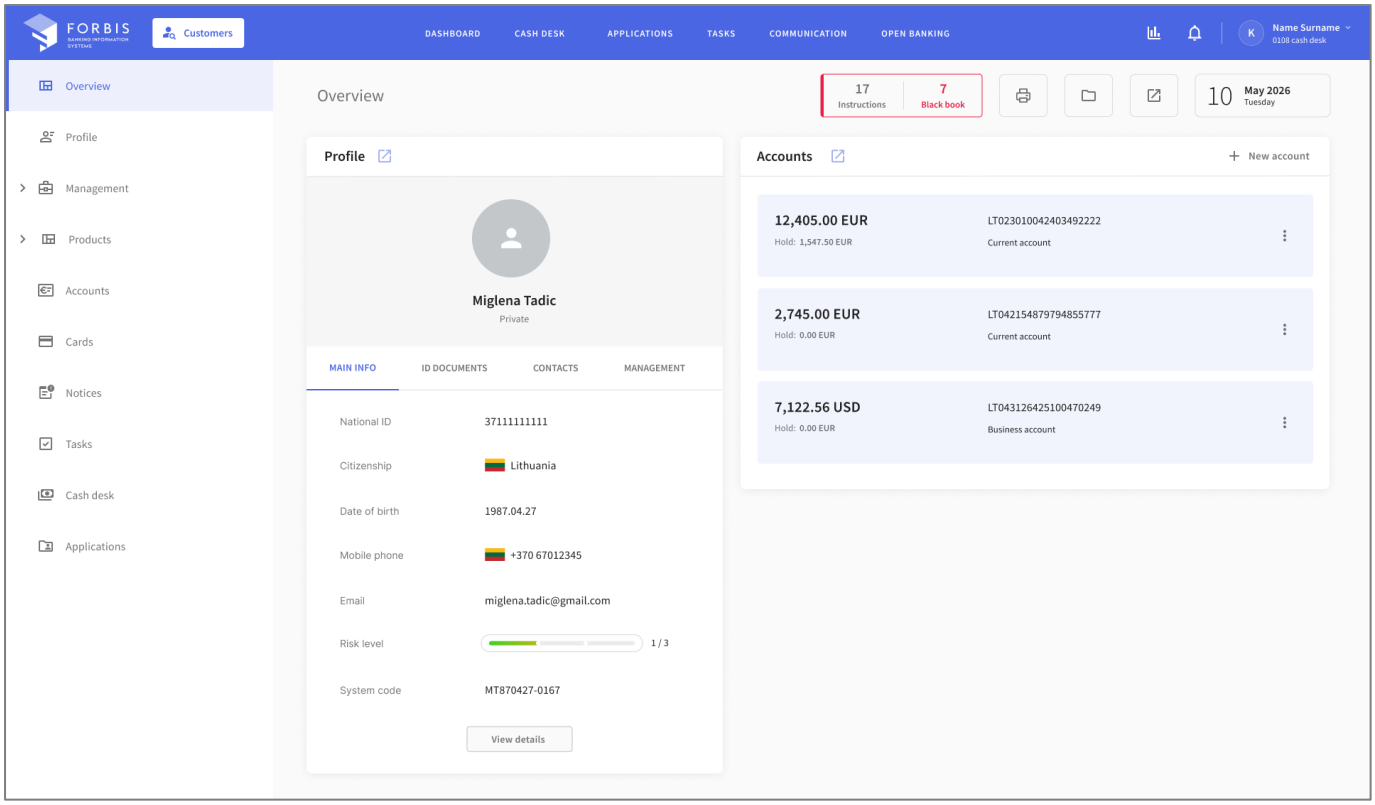
The application provides:

- intuitive workflows aligned with users' mental models,
- clear visual hierarchy and information grouping,
- consistent feedback mechanisms and status visibility,
- optimized layouts for data-intensive operations,
- reduced cognitive load during complex analytical and administrative tasks.

The interface design supports both experienced operational users and less frequent administrative users by ensuring predictable interaction behaviour and simplified task flows.

Visual design decisions prioritize clarity and usability over decorative elements, supporting efficient long-term daily use within a professional financial environment.





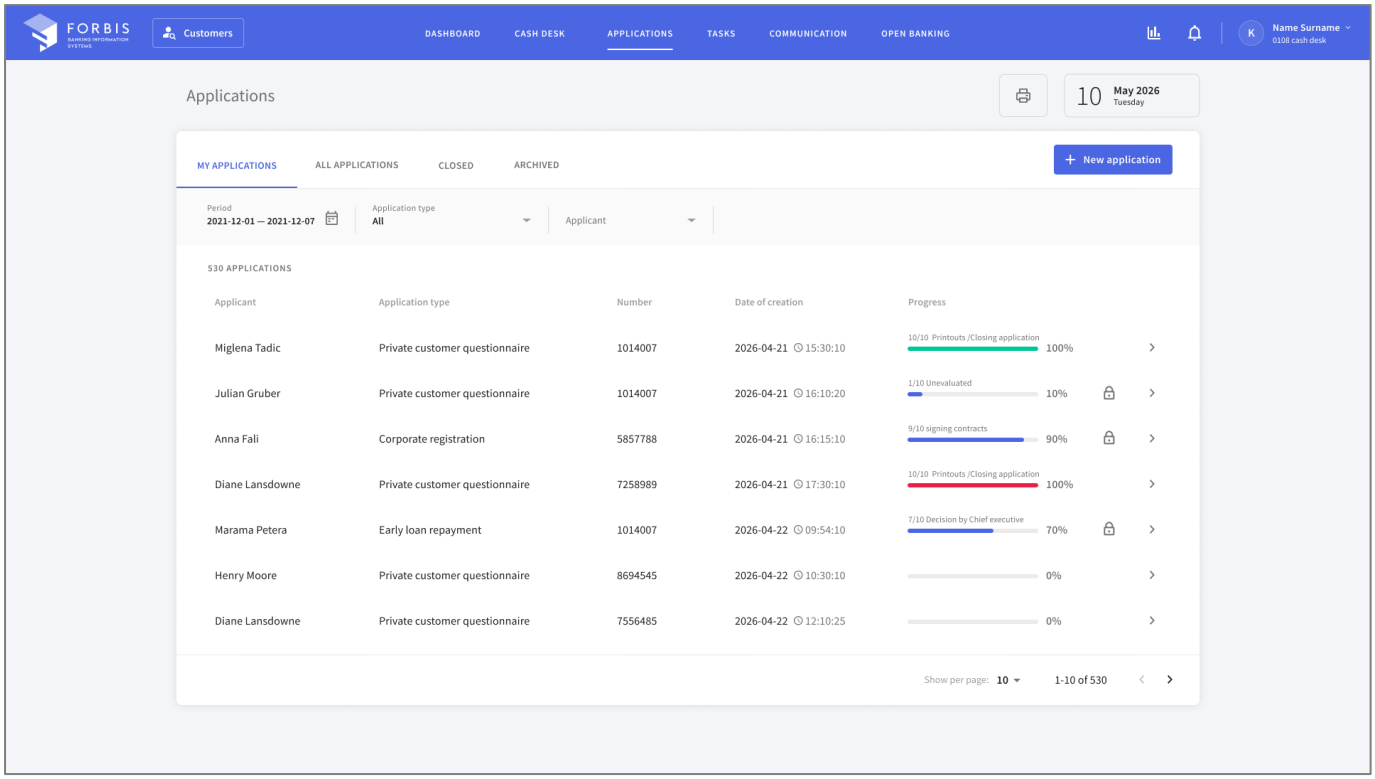
The application follows a centralized design system and unified graphical language to ensure visual and functional consistency across all modules and workflows.

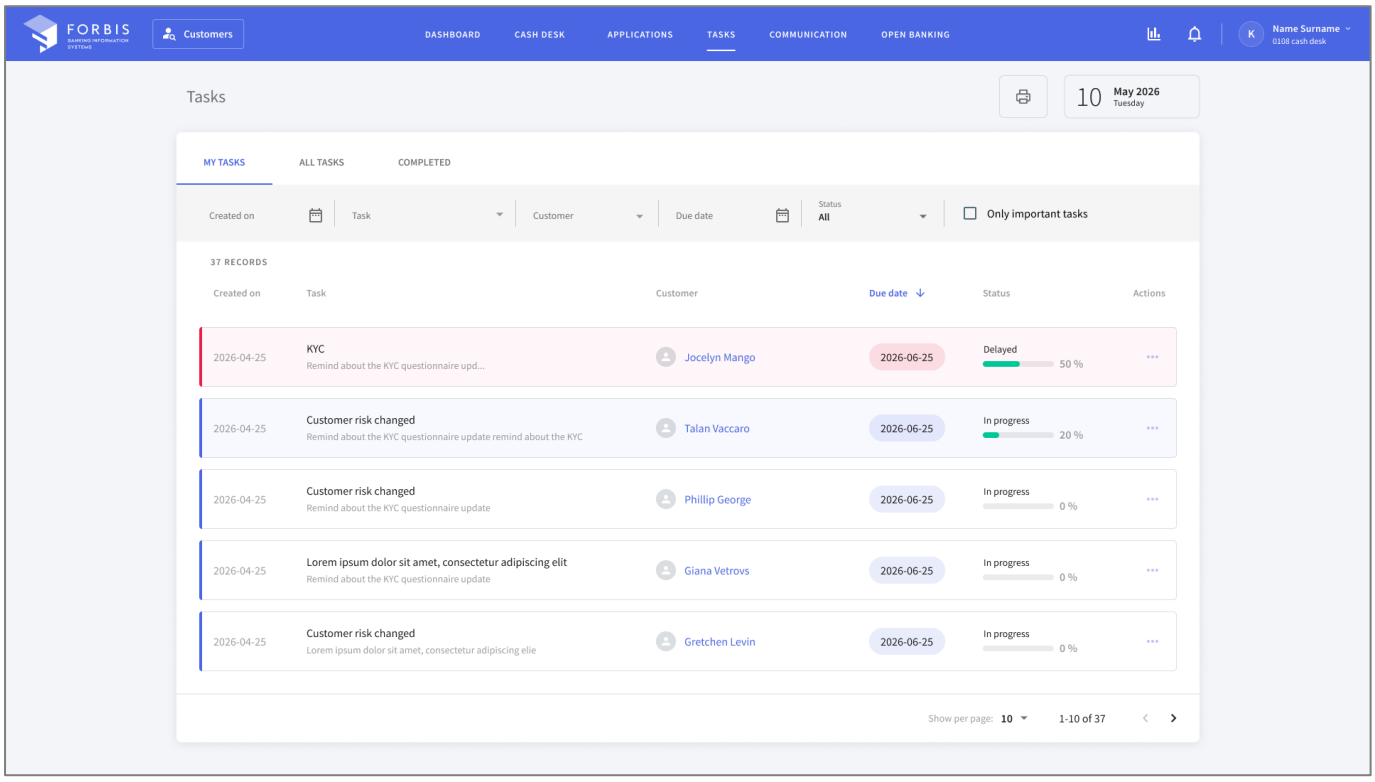
Reusable interface components, typography rules, spacing systems, colour semantics, icons, tables, forms, and interaction states are standardized throughout the application. This consistency improves learnability, reduces training effort, and minimizes the likelihood of user errors during daily operations.

The design system also ensures:

- consistent handling of alerts, warnings, confirmations, and validation states,
- unified behaviour of forms and data tables,
- predictable placement of actions and controls,
- accessibility and readability across different user roles and screen sizes.

The consistent user experience significantly reduces cognitive effort for users working with complex regulatory and operational processes.





The application supports multiple complementary interaction mechanisms to ensure efficient and accessible navigation throughout the system.

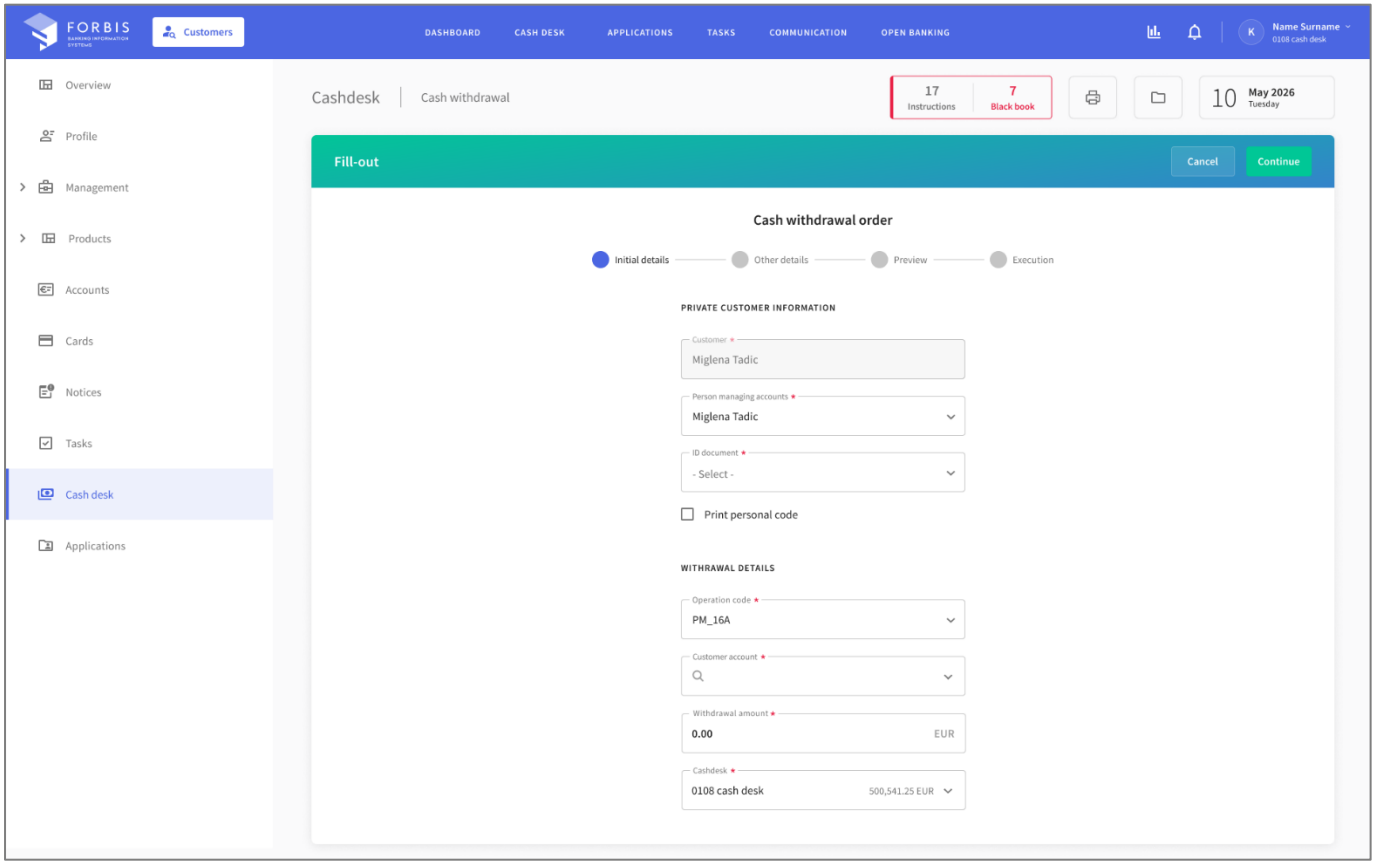
Users are able to interact with the application using:

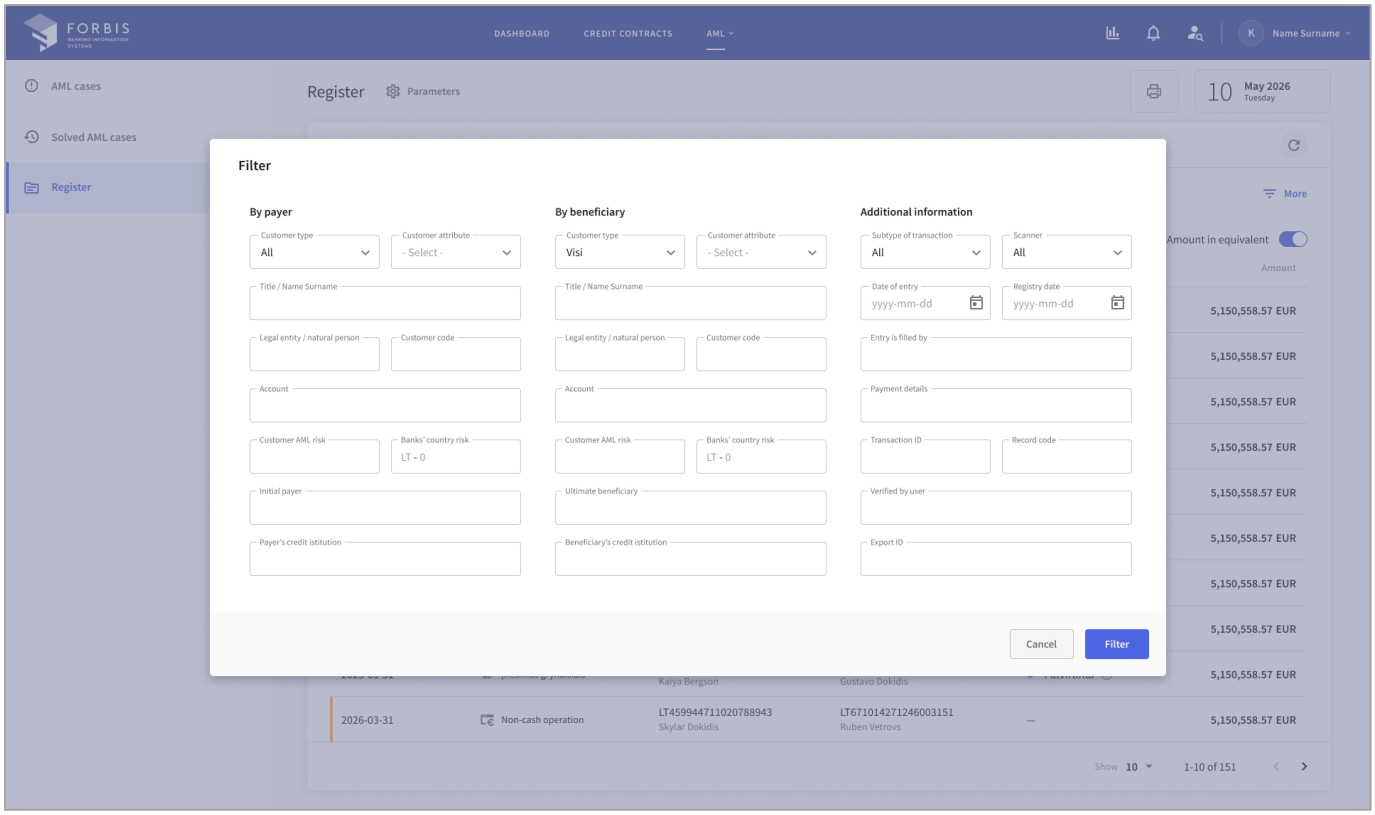
- mouse-based interactions,

- keyboard navigation,
- contextual actions,
- filtering and search mechanisms,
- expandable side panels and modal workflows where appropriate.

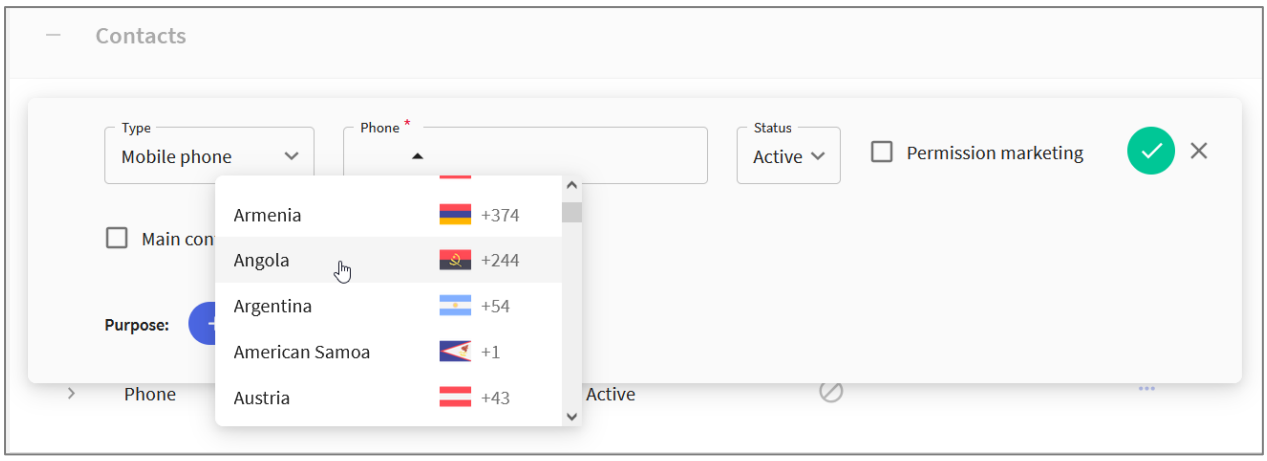
Special attention is given to minimizing unnecessary context switching between keyboard and mouse interactions during high-frequency operational tasks.

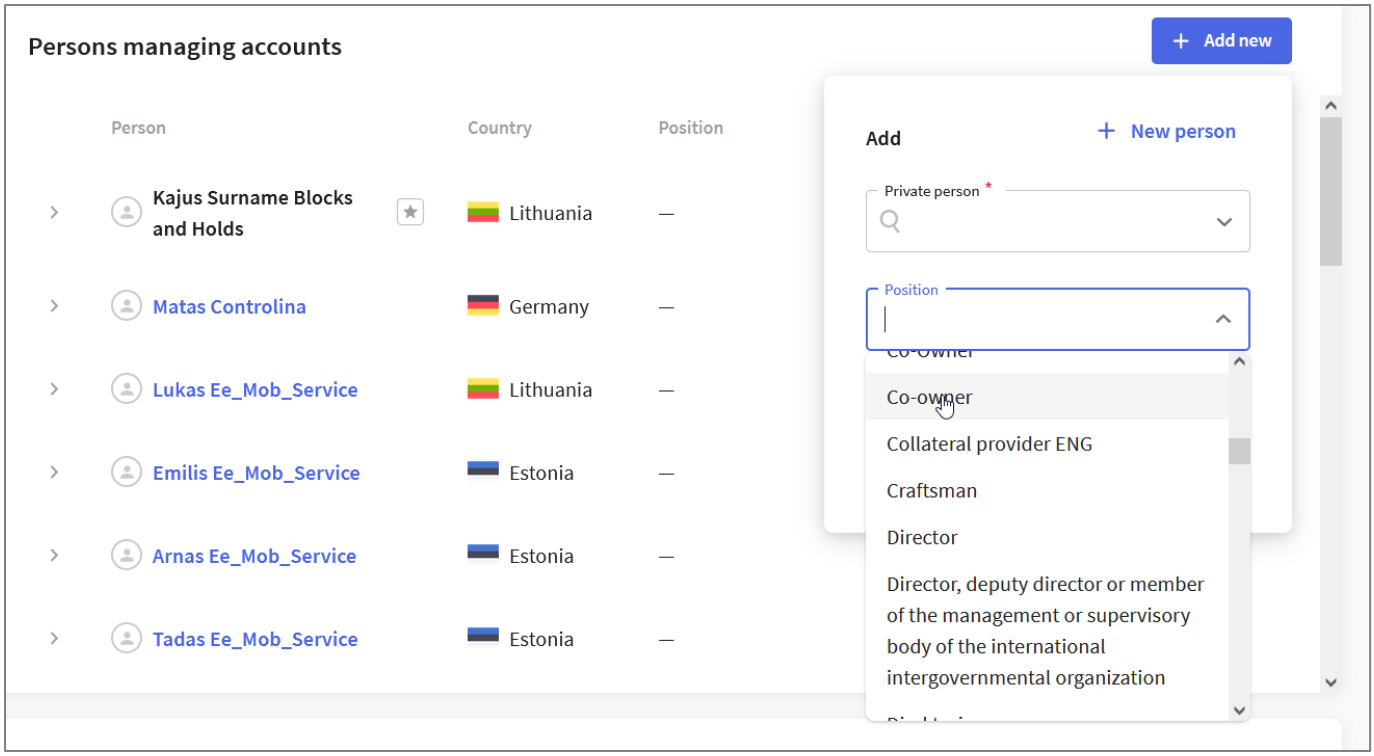
The interface supports efficient movement between forms, records, and analytical views while preserving user context and reducing repetitive actions. Navigation patterns are designed to remain consistent across all modules to improve operational speed and reduce training requirements.



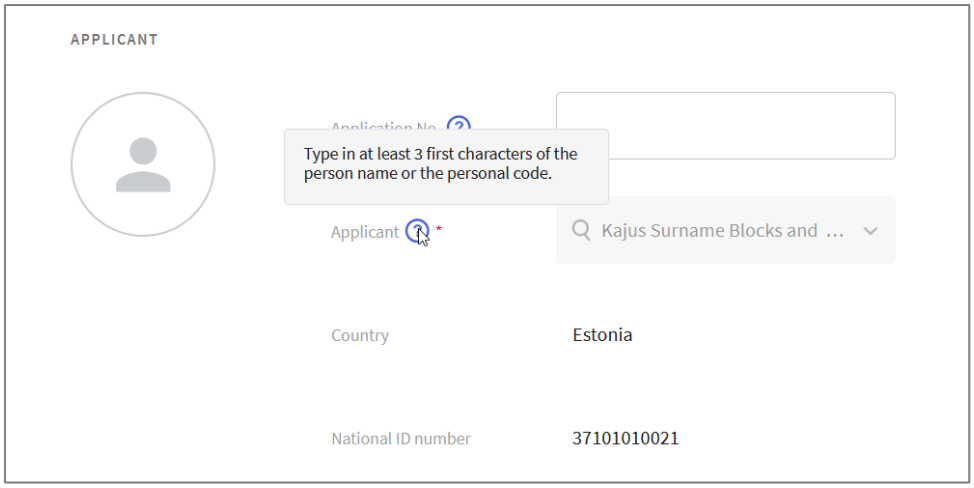


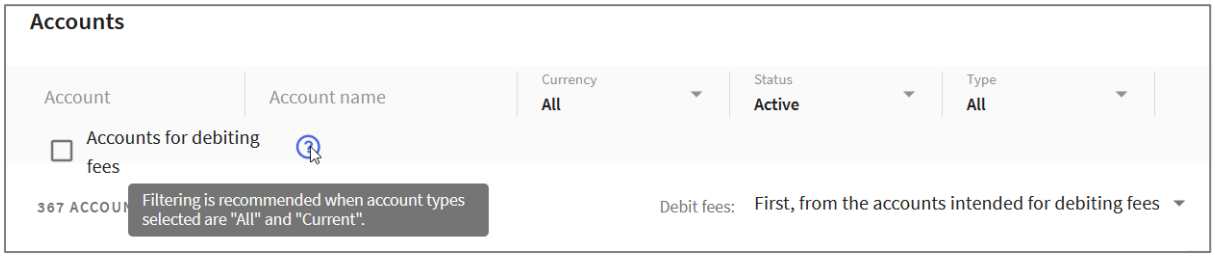
Wherever possible, the user is offered lists of available values for various input fields:



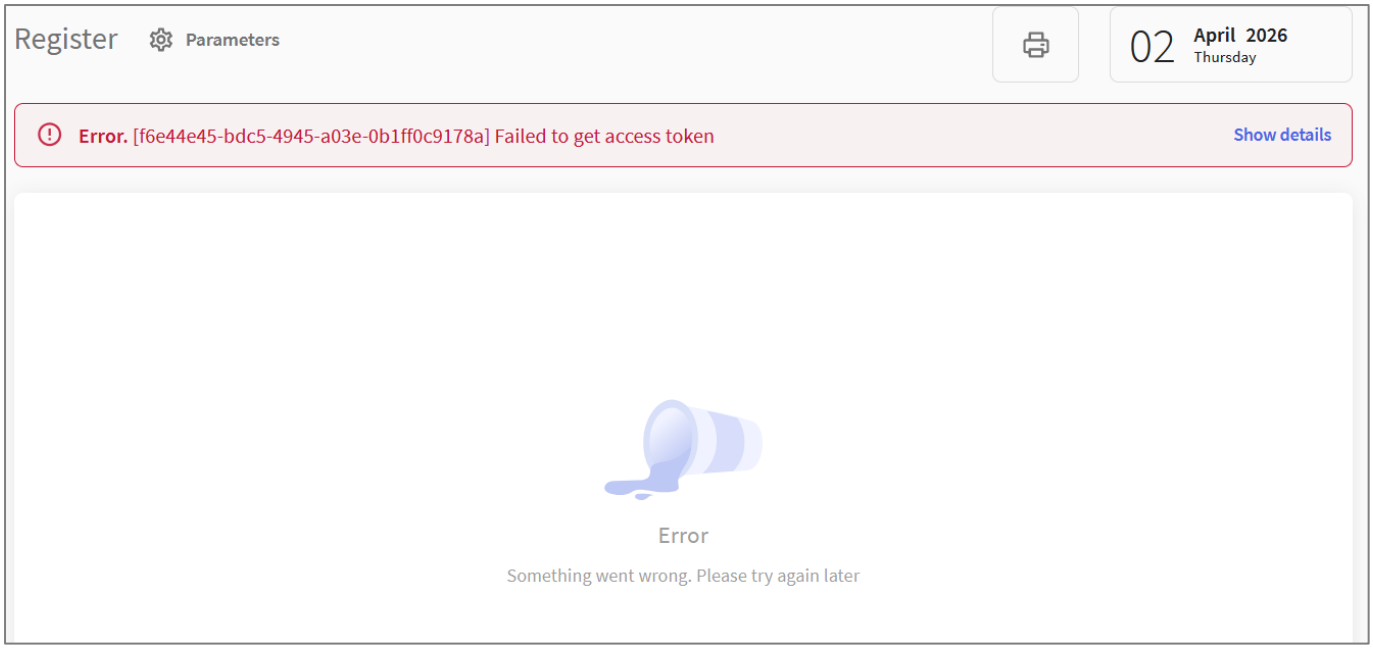


The user is also offered context-sensitive hints (an icon with a question mark appears next to the field):

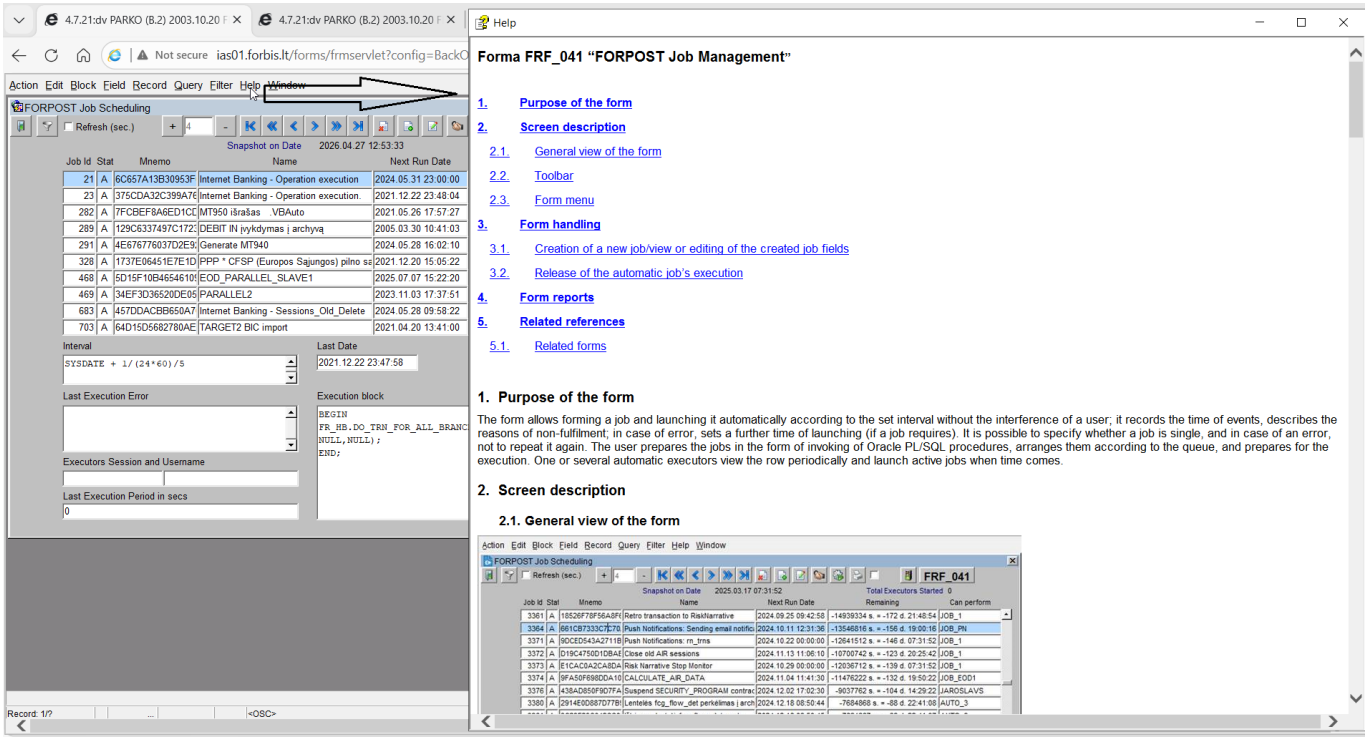




In case of an error, application displays user-friendly error message:

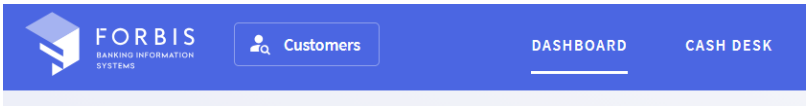


13.1.1 Detailed help files in forms for back-office administrators

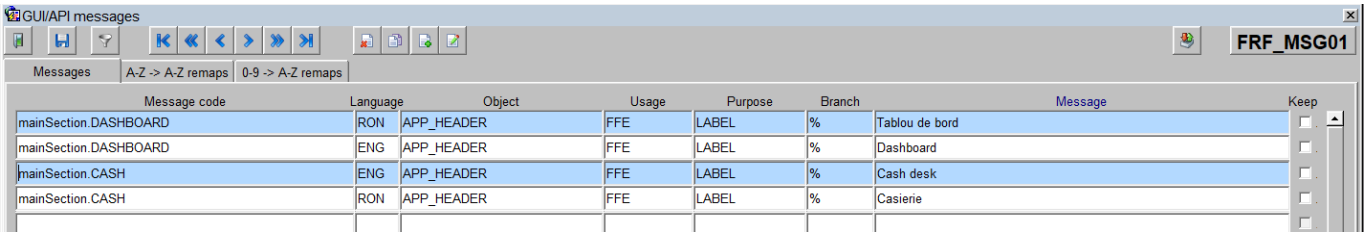


13.2 Interface localization (CNF.127, CNF.128)

Interface localization is supported at the kernel and interface levels. All messages are stored in the kernel and can be exported as separate files. Messages can be used for forms, reports, and kernel packages. For example: front-end application menu in English:

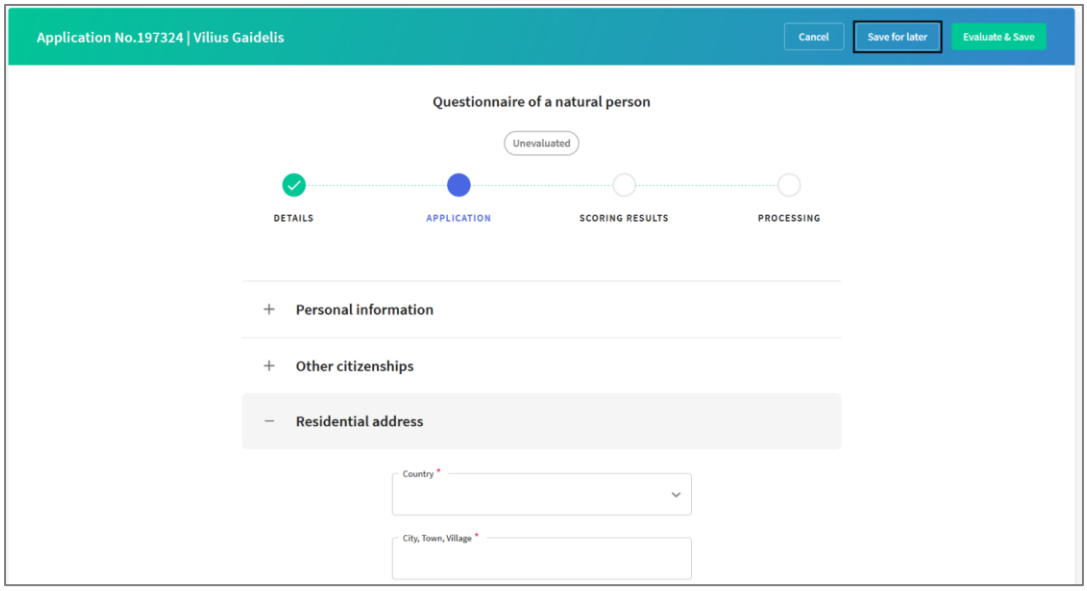


Texts in translation form:

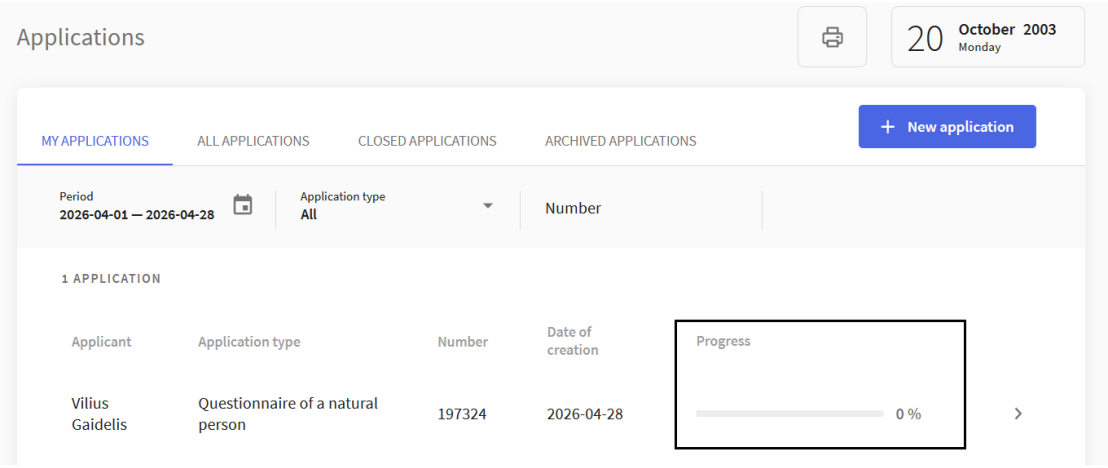


13.3 Resume-later support (CNF.126)

When filling out complex forms (for example, various questionnaires), the user is given the opportunity to interrupt the filling process early (while saving the intermediate result) in order to return to filling it out later:



The user also has the ability to see their progress in filling out a complex form:



14. Conclusion

The proposed solution is designed as a flexible, modular, secure, scalable, and enterprise-grade solution supporting NBM's banking operations, integration needs, Remote Banking services, data management, security, monitoring, and operational administration. The solution architecture provides controlled processing, data exchange, validation, auditability, logging, access control, maintainability, and operational governance required for a critical central-bank environment.

The proposed solution is technology-flexible and may be deployed using enterprise-supported Rancher RKE2 products. Red Hat products or Oracle Cloud Infrastructure (OCI) components are optional, depending on the final technological platform selected by NBM. In addition, the proposed architecture remains open to other enterprise Kubernetes platforms, subject to NBM's technology standards, security requirements, and final deployment design. This gives NBM the ability to choose the most suitable platform based on its infrastructure strategy, security requirements, licensing model, operational capabilities, resilience expectations, maintainability, and total cost of ownership.

Regardless of the selected platform, the proposed solution will follow the same architectural principles: open standards, modular architecture, secure communication, scalability, high availability, centralized monitoring, auditability, controlled administration, and long-term maintainability. This ensures that the solution remains aligned with NBM's current and future business, security, interoperability, and operational requirements.

Appendix A. Infrastructure Sizing and Deployment

This appendix defines the infrastructure sizing, deployment topology and environment separation principles for the proposed solution. It is intended to support evaluation of platform compatibility, operational readiness, scalability, maintainability and deployment feasibility.

The proposed deployment is based on standard market-available x86 infrastructure, compatible with a virtualized VMware environment, and organized into separate network zones for client-facing services, business system DMZ services, internal applications services and database services. The production topology uses redundant applications and service nodes wherever the component is operationally critical, with traffic routed through load balancers and state maintained in the database and controlled application repositories.

A.1 Deployment zones and component placement

Zone	Main Components	Deployment rationale
Client DMZ	Remote banking web front-end, web API services, WAF/load balancer and customer-facing access controls.	Separates externally reachable services from internal banking services. Only controlled TLS/HTTPS flows are allowed towards internal APIs and integration services.
Business System DMZ	Reporting services, external service adapters, integration services, message broker, trust store and key store components.	Provides a controlled mediation layer for external and inter-system communication. Enables protocol adaptation, message validation, routing and operational isolation.
Internal zone	Back-office Oracle Forms services, front-office/middle-office API services, authentication services and internal load balancing.	Hosts business user channels and application services that interact with database APIs through controlled JDBC and service interfaces.
DB zone	Oracle database, RAC repository, local standby and secure database access interfaces.	Serves as the authoritative system of record, central business rules execution point, audit source and consistency boundary.

Zone	Main Components	Deployment rationale
Disaster recovery site	Remote standby database and recovery instances for critical application services.	Supports active-passive continuity across two geosites, with recovery procedures designed around the agreed RTO, RPO and switchover objectives.

A.2 Indicative production sizing (CNF.54, CNF.55, CNF.56, CNF.57)

The sizing below is an initial baseline for supplier evaluation and infrastructure planning. Final sizing will be confirmed during the detailed design phase using agreed user volumes, transaction profiles, reporting loads, integration throughput, retention requirements and resilience targets.

RKE2-based solution (suggested, included in the Tender pricing):

Component	Purpose	Technology stack	Indicative production deployment
RAC: FBS Database / Oracle Forms Repository	System kernel database and Oracle Forms repository; central data storage and high availability through Oracle RAC and TAF.	Oracle Database 19c Enterprise Edition Certified enterprise Linux x86_64 platform, subject to Oracle support matrix and NBM standards.	2 RAC nodes, 16 CPU cores per node, 256 GB RAM total, 4 TB SSD.
Local Standby	Local standby database for local resilience and recovery support.	Oracle Database 19c Enterprise Edition.	32 CPU cores, 256 GB RAM, 4 TB SSD.
Remote Standby	Remote standby database for disaster recovery site.	Oracle Database 19c Enterprise Edition with Data Guard/Active Data Guard approach.	32 CPU cores, 256 GB RAM, 4 TB SSD.
Oracle 14 Forms Server	Secure back-office console for internal administrative and operational banking activities.	Windows Server 2019 x64 Oracle Forms 14c WebLogic Server Fusion Middleware 14c.	2 server nodes, each with 4 CPU cores, 32 GB RAM, 200 GB SSD/NVMe.
Rancher RKE2 Kubernetes platform	Core container orchestration platform for middleware, API, authentication, reporting, and Internet banking workloads; provides scheduling, service discovery, container runtime, ingress integration, and network-policy enforcement.	Rancher RKE2 Kubernetes; HA RKE2 server/control-plane nodes; RKE2 agent/worker nodes; RKE2 NGINX Ingress Controller; Private registry and monitoring/logging integrations as agreed with NBM.	Recommended minimum: 3 RKE2 server/control-plane nodes for HA plus dedicated worker node pools for Internal and Client DMZ workloads. Final CPU/RAM/storage sizing to be confirmed during detailed design based on workload profiles and HA requirements.
Internal Load Balancer	Balances internal user and service traffic between redundant application instances and exposes internal RKE2 ingress endpoints.	Corporate L4/L7 load balancer integrated with RKE2 NGINX Ingress Controller; Optional HA Proxy/F5/NSX Advanced Load Balancer or equivalent NBM-approved solution.	4 CPU cores, 8 GB RAM, 300 GB SSD/NVMe.
Front-office and middle-office services	Browser-based internal banking consoles and API services for front-office and middle-office users.	React-based browser user interface; Rancher RKE2/Kubernetes; Apache HTTP Server; JBoss EAP or WildFly application runtime; OpenJDK 21; Oracle client. Client Side: • Microsoft Edge (latest 3 versions) • Google Chrome (latest 3 versions) • Mozilla Firefox (latest 3 versions) • Apple Safari (latest 3 versions)	2 server nodes, each with 16 CPU cores, 32 GB RAM, 200 GB SSD/NVMe.
Jasper reporting services	PDF and other report generation services.	Rancher RKE2/Kubernetes;	2 server nodes, each with 8 CPU cores, 32 GB RAM, 300 GB SSD/NVMe.

Component	Purpose	Technology stack	Indicative production deployment
		JBoss EAP or WildFly application runtime; OpenJDK; JasperReports Server/Library; Oracle client.	
MS 365 report services	Generation of Microsoft Word and Microsoft Excel report outputs.	Adjacent Windows Server 2022 service integrated with RKE2 workloads over TLS APIs; Microsoft Office/Word/Excel; OpenJDK 8; Spring Boot.	2 server nodes, each with 2 CPU cores, 8 GB RAM, 100 GB HDD/SSD.
Authentication server	Centralized authentication and single sign-on for front-office and middle-office channels.	Rancher RKE2/Kubernetes; Keycloak containerized deployment; OpenJDK 21; External HA database for Keycloak state; TLS ingress through RKE2 ingress and corporate load balancer.	2 server nodes, each with 4 CPU cores, 8 GB RAM, 100 GB SSD.
Integration and external services	Secure data exchange with external systems and supporting integration functions such as transformation and HSM connectivity.	Rancher RKE2/Kubernetes; JBoss EAP or WildFly application runtime; OpenJDK 17; Spring Boot; Apache Camel; Apache ActiveMQ Artemis or NBM-approved enterprise messaging broker; Secure HSM connectivity.	2 server nodes, each with 8 CPU cores, 32 GB RAM, 300 GB SSD/NVMe.
Client DMZ Load Balancer	Balances customer-facing channel traffic and exposes Internet banking services through the approved security boundary.	Corporate WAF/load balancer integrated with RKE2 NGINX Ingress Controller; Dedicated RKE2 Client DMZ worker pool or separate DMZ RKE2 cluster depending on NBM security policy.	4 CPU cores, 8 GB RAM, 300 GB SSD/NVMe if implemented as virtual appliances.
Remote banking services	Customer web channel for secure access to banking products and services.	React-based browser user interface; Rancher RKE2/Kubernetes; Apache HTTP Server; JBoss EAP or WildFly application runtime; OpenJDK 17; WAF/load balancer; Anti-malware scanning. Client Side: • Microsoft Edge (latest 3 versions) • Google Chrome (latest 3 versions) • Mozilla Firefox (latest 3 versions) • Apple Safari (latest 3 versions)	2 server nodes, each with 12 CPU cores, 32 GB RAM, 300 GB SSD/NVMe.

Red Hat-based solution (optional, not included in the price):

Component	Purpose	Technology stack	Indicative production deployment
RAC: FBS Database / Oracle Forms Repository	System kernel database and Oracle Forms repository; central data storage and high availability through Oracle RAC and TAF.	Oracle Database 19c Enterprise Edition; Red Hat Enterprise Linux.	2 RAC nodes, 16 CPU cores per node, 256 GB RAM total, 4 TB SSD.
Local Standby	Local standby database for local resilience and recovery support.	Oracle Database 19c Enterprise Edition.	32 CPU cores, 256 GB RAM, 4 TB SSD.
Remote Standby	Remote standby database for disaster recovery site.	Oracle Database 19c Enterprise Edition with Data Guard/Active Data Guard approach.	32 CPU cores, 256 GB RAM, 4 TB SSD.
Oracle 14 Forms Server	Secure back-office console for internal administrative and operational banking activities.	Windows Server 2019 x64; Oracle Forms 14c; WebLogic Server Fusion Middleware 14c.	2 server nodes, each with 4 CPU cores, 32 GB RAM, 200 GB SSD/NVMe.

Component	Purpose	Technology stack	Indicative production deployment
Internal Load Balancer	Balances internal user and service traffic between redundant application instances.	Red Hat OpenShift HTTP Balancer / corporate load balancing.	4 CPU cores, 8 GB RAM, 300 GB SSD/NVMe.
Front-office and middle-office services	Browser-based internal banking consoles and API services for front-office and middle-office users.	React-based browser user interface; Red Hat OpenShift/Kubernetes; JBoss EAP; OpenJDK 21; Apache HTTP Server; Oracle client. Client Side: • Microsoft Edge (latest 3 versions) • Google Chrome (latest 3 versions) • Mozilla Firefox (latest 3 versions) • Apple Safari (latest 3 versions)	2 server nodes, each with 16 CPU cores, 32 GB RAM, 200 GB SSD/NVMe.
Jasper reporting services	PDF and other report generation services.	Red Hat OpenShift; JBoss EAP; OpenJDK; JasperReports Server/Library; Oracle client.	2 server nodes, each with 8 CPU cores, 32 GB RAM, 300 GB SSD/NVMe.
MS 365 report services	Generation of Microsoft Word and Microsoft Excel report outputs.	Windows Server 2022 Microsoft Office/Word/Excel OpenJDK 8 Spring Boot.	2 server nodes, each with 2 CPU cores, 8 GB RAM, 100 GB HDD/SSD.
Authentication server	Centralized authentication and single sign-on for front-office and middle-office channels.	Red Hat OpenShift/Kubernetes; Red Hat build of Keycloak; OpenJDK 21.	2 server nodes, each with 4 CPU cores, 8 GB RAM, 100 GB SSD.
Integration and external services	Secure data exchange with external systems and supporting integration functions such as transformation and HSM connectivity.	Red Hat OpenShift; JBoss EAP; OpenJDK 17; Spring Boot; Apache Camel; Red Hat AMQ Broker.	2 server nodes, each with 8 CPU cores, 32 GB RAM, 300 GB SSD/NVMe.
Client DMZ Load Balancer	Balances customer-facing channel traffic.	Red Hat OpenShift HTTP Balancer / corporate load balancing.	4 CPU cores, 8 GB RAM, 300 GB SSD/NVMe.
Remote banking services	Customer web channel for secure access to banking products and services.	React-based browser user interface; Red Hat OpenShift; JBoss EAP; Apache HTTP Server; OpenJDK 17; WAF/load balancer; Anti-malware scanning. Client Side: • Microsoft Edge (latest 3 versions) • Google Chrome (latest 3 versions) • Mozilla Firefox (latest 3 versions) • Apple Safari (latest 3 versions)	2 server nodes, each with 12 CPU cores, 32 GB RAM, 300 GB SSD/NVMe.

Oracle Cloud Infrastructure (OCI)- based solution (optional, not included in the price):

Component	Purpose	Technology stack	Indicative production deployment
FBS Database / Oracle Forms Repository	System kernel database and Oracle Forms repository; central data storage and high availability.	Oracle Autonomous Database on OCI with Oracle-managed backup and recovery. Disaster recovery can be enhanced using Autonomous Data Guard or backup-based disaster recovery,	Autonomous Database sized according to workload, storage volume, number of users, and performance requirements. Final OCPU/storage configuration, backup retention, and DR model to be agreed with NBM during implementation planning.

Component	Purpose	Technology stack	Indicative production deployment
		depending on NBM RTO/RPO requirements.	
Internal Load Balancer	Balances internal user and service traffic between redundant application instances.	OCI Load Balancer or OCI Network Load Balancer for internal private traffic.	Private load balancer with redundant backend pools for Forms, application services, and internal APIs.
Oracle 14 Forms Server	Secure back-office console for internal administrative and operational banking activities.	Windows Server 2019 x64 Oracle Forms 14c WebLogic Server Fusion Middleware 14c.	2 server nodes, each with 4 CPU cores, 32 GB RAM, 200 GB SSD/NVMe.
Front-office, middle-office services	Browser-based internal banking consoles and API services for front-office, middle-office and back-office users.	React-based browser user interface, Container Orchestration: Oracle OKE Application Runtime: OKE JBoss Enterprise Application Platform (JBoss EAP) OpenJDK 21 Web Tier: Apache HTTP Server deployed on Oracle OKE. Database Connectivity: Oracle Database 19c Client (OJDBC) OCI Wallet OCI Vault Client Side: Microsoft Edge (latest 3 versions) Google Chrome (latest 3 versions) Mozilla Firefox (latest 3 versions) Apple Safari (latest 3 versions) Network / Access Layer: OCI Load Balancer	OKE worker nodes equivalent to current sizing: 2 worker nodes, approximately 16 OCPU and 32 GB RAM each, with OCI Block Volume as required.
Jasper reporting services	PDF and other report generation services.	Container Orchestration: Oracle OKE Application Runtime: Apache Tomcat container image OpenJDK Reporting Platform: JasperReports Server JasperReports Library Database Connectivity: Oracle Database 19c Client (OJDBC, including NLS support and Advanced Queueing support where applicable) OCI Wallet OCI Vault	2 OKE worker nodes or Compute VMs, each approximately 8 OCPU, 32 GB RAM, 300 GB OCI Block Volume.
MS 365 report services	Generation of Microsoft Word and Microsoft Excel report outputs.	OCI Compute VM Windows Server 2022 Microsoft Office/Word/Excel where licensing permits OpenJDK 8 Spring Boot	2 OCI Compute VM instances, each approximately 2 OCPU, 8 GB RAM, 100 GB OCI Block Volume.
Authentication server	Centralized authentication and single sign-on for front-office and middle-office channels.	Oracle OKE OCI Wallet OCI Vault Keycloak deployed on Oracle OKE OpenJDK 21	2 OKE pods/nodes each approximately 4 OCPU, 8 GB RAM, 100 GB storage.
Integration and external services	Secure data exchange with external systems and supporting	Oracle OKE OCI Wallet OCI Vault	2 OKE worker nodes or Compute VMs, each approximately 8 OCPU, 32 GB RAM, 300 GB OCI Block Volume.

Component	Purpose	Technology stack	Indicative production deployment
	integration functions such as transformation and HSM connectivity.	OpenJDK 17 Apache Tomcat container image OCI Queue / Streaming where applicable, or AMQ on OCI Compute/OKE Spring Boot Framework	
Client DMZ Load Balancer	Balances customer-facing channel traffic.	OCI Public Load Balancer with WAF protection where required.	Public load balancer with frontend listener, backend pools, TLS configuration, and optional OCI WAF.
Remote banking services	Customer web channel for secure access to banking products and services.	React-based browser user interface, Oracle OKE OCI Wallet OCI Vault OpenJDK 17 JBoss Enterprise Application Platform Oracle Apache HTTP Server Oracle Web Application Firewall (WAF) Oracle Load Balancer Anti-Malware / File Scanning Service(CLAMAV) Client Side: • Microsoft Edge (latest 3 versions) • Google Chrome (latest 3 versions) • Mozilla Firefox (latest 3 versions) • Apple Safari (latest 3 versions)	2 OKE worker nodes, each approximately 12 OCPU, 32 GB RAM, 300 GB OCI Block Volume.

In the OCI-based alternative, the database layer may be implemented using Oracle Autonomous Database, reducing the need for customer-managed RAC, local standby, and remote standby configurations. Backup, recovery, availability, and disaster recovery capabilities are provided through OCI-native Autonomous Database services. The final recovery configuration, including the need for Autonomous Data Guard, cross-region disaster recovery, or backup-based recovery, shall be defined together with NBM based on confirmed RTO/RPO, regulatory, and operational requirements.

A.3 Environment separation

The project will apply controlled environment separation across development, quality assurance, integration testing, pre-production and production. The non-production environments will replicate the functional architecture of production while using reduced infrastructure capacity where appropriate. Data transfer between environments will be controlled through approved procedures, and any production-derived data used outside production will be depersonalized or masked in accordance with agreed security rules.

Environment	Purpose	Controls
Development / CI	Supplier development, automated build, code inspection, vulnerability scanning and unit-level validation.	Access restricted to project team Build artefacts versioned Changes traceable to backlog and change records.
Test / Integration	Functional, interface, migration rehearsal and regression testing.	Controlled test data Integration endpoints simulated or connected through agreed test channels Test results retained
Pre-production	Operational rehearsal, acceptance testing, performance validation and go-live preparation.	Configuration aligned with production Access and change control aligned with deployment procedures
Production	Live banking service operation.	Restricted privileged access Formal change approval Monitoring Backup

Environment	Purpose	Controls
		Disaster recovery and incident procedures.
Disaster recovery	Recovery of critical services during site-level incident or continuity test.	Data replication Recovery runbooks Switchover rehearsal and documented return-to-normal procedures

A.5 Deployment and release prerequisites

- Infrastructure capacity, VM templates, network segmentation, DNS, certificates and load-balancer rules are confirmed before component deployment.
- Database schemas, applications configuration, secrets, trust stores and key stores are prepared through controlled installation procedures.
- Kubernetes projects/namespaces, service accounts, routes and resource limits are prepared for containerized workloads.
- Oracle Forms/WebLogic and Windows-based reporting services are installed in accordance with approved platform hardening rules.
- Rollback and back-out procedures are prepared for every production deployment, including database change sequencing and applications compatibility checks.

Appendix B. Security and IAM (CNF.136, CNF.137, CNF.139, CNF.141)

This appendix describes the security and identity and access management architecture of the proposed solution. The security model is based on layered controls: network segmentation, secure transport, strong authentication, role-based authorization, controlled privileged access, auditability, application level validation and protection of data throughout its lifecycle.

B.1 Zero Trust aligned security model

The architecture follows a Zero Trust aligned approach: no network zone or user context is implicitly trusted, and all access to business functions, APIs and administrative capabilities is subject to authentication, authorization, traffic control and logging. The model combines perimeter controls with service-level and database-level controls, so that the compromise or failure of one layer does not automatically provide access to sensitive functions or data.

Security domain	Control implementation	ISO/IEC 27001 / 27002 alignment rationale
Network segmentation	Client DMZ, Business System DMZ, Internal and DB zones; restricted traffic flows through load balancers, WAF, APIs and JDBC.	Supports network security and segregation of networks by separating systems into controlled security zones and limiting traffic between external, applications and database layers. Relevant to controls such as A.8.20 Network security, A.8.21 Security of network services and A.8.22 Segregation of networks.
Secure communication	TLS/HTTPS for service and user traffic; controlled JDBC to Oracle; no direct external access to internal components.	Supports secure information transfer and cryptographic protection of data in transit. It also reduces exposure of internal components by ensuring that communication paths are controlled and encrypted. Relevant to A.5.14 Information transfer, A.8.20 Network security, A.8.21 Security of network services and A.8.24 Use of cryptography.
Identity and authentication	Keycloak for modern web channels with MFA/OTP capabilities; Oracle authentication for back-office administration with MFA capability.	Supports controlled identity management and secure authentication by using centralized identity services and MFA-capable authentication for user and administrative access. Relevant to A.5.16 Identity management, A.5.17 Authentication information, A.8.5 Secure authentication and A.8.2 Privileged access rights.
Authorization	Role-based and group-based permissions; business-layer roles control functions, data and operations.	Supports access control and least-privilege authorization by ensuring that users can access only the functions, data and operations assigned to their role or group. Relevant to A.5.15 Access control, A.5.18 Access rights, A.8.2 Privileged access rights and A.8.3 Information access restriction.
Credential protection	No open-text storage of secrets; secure configuration and restricted administration.	Supports protection of authentication information and secure configuration management by preventing clear-text secret

Security domain	Control implementation	ISO/IEC 27001 / 27002 alignment rationale
		storage and limiting administrative access to sensitive configuration. Relevant to A.5.17 Authentication information, A.8.5 Secure authentication, A.8.9 Configuration management and A.8.24 Use of cryptography.
Input/output validation	Client-side assistance plus authoritative validation in Oracle business APIs and business procedures.	Supports secure applications processing by ensuring that validation is enforced on the trusted server/business layer, not only in the client interface. Relevant to A.8.26 Application security requirements, A.8.28 Secure coding and A.8.29 Security testing in development and acceptance.
Data confidentiality	Role-based access, masking/protection for sensitive data where agreed, secure report and export controls.	Supports confidentiality of information assets by restricting access to sensitive data, applying masking/protection where required, and controlling report/export channels. Relevant to A.5.12 Classification of information, A.5.15 Access control, A.8.3 Information access restriction, A.8.11 Data masking and A.8.12 Data leakage prevention.

B.2 Identity, roles and access lifecycle (CNF.135, CNF.147, CNF.148, CNF.149, CNF.150, CNF.151, CNF.152)

User access is assigned through controlled roles and user groups. Each user group is mapped to business functions, interface capabilities, data visibility and administrative permissions. The front-office and middle-office web applications use Keycloak-based authentication and session handling, while back-office administrative functions are controlled through Oracle Forms and business-layer roles. Internet banking authentication will be integrated with the authenticator selected for the Moldovan context, such as e-signature or an equivalent national authentication mechanism.

- Access is granted only through approved roles and groups aligned with the user's business responsibility.
- Administrative access is separated from operational access and limited to authorized administrators.
- Access changes are logged and subject to the agreed approval process.
- Critical business processes can apply maker-checker or four-eyes control where specified in the detailed design.
- Sessions, tokens and credentials are protected through secure configuration and transport encryption.

MFA support

The solution supports multifactor authentication for user access as part of the overall identity and access management model. For front-office and middle-office applications, MFA is implemented using standard Keycloak capabilities, including configurable

authentication flows and policy-based enforcement. This allows MFA to be applied according to user group, role, access channel, and security requirements. For back-office components based on Oracle Forms, the solution provides a dedicated MFA

mechanism aligned with the back-office authentication model. This ensures that users accessing back office functionality are subject to an additional authentication factor in addition to standard credentials.

Password storage and protection

The solution applies a common password-protection mechanism across all application components, including back-office, middle-office, front-office, and related services. Application user passwords are not stored in plain text and are protected using. Oracle

based password verifier mechanisms, including the 12C password version where applicable. The password mechanism is based on one-way cryptographic protection, meaning that passwords cannot be recovered from the stored verifier value. Authentication

is performed by validating the user-provided password against the protected verifier. Password recovery is not supported, where required passwords are changed through controlled password reset procedures. The same password-protection approach is

applied consistently across the system, regardless of the user interface technology used by individual components. Front-end applications, such as web interfaces, do not store or expose passwords and perform authentication only through secure backend

authentication services. Password transmission is protected using HTTPS/TLS. Access to authentication data and password-related configuration is restricted to authorized administrators and is subject to audit logging and security monitoring.

Authentication and Password Policy Management

The System provides configurable password policy management for application users. Password policies may be enforced either through integration with the centralized NBM authentication system or independently by the System for users that are not

integrated with centralized authentication.

Policy area	System capability
Password complexity requirements	Configuration of password strength rules, such as minimum length, character types, and complexity requirements.
Mandatory password change	Ability to require password change after first login, after password reset, or according to administrative/security policy.
Password expiration policies	Ability to define password validity periods and enforce password renewal after expiration.
Prevention of password reuse	Ability to prevent users from reusing previously used passwords according to the configured password history policy.
Failed authentication limits	Configurable limits for failed login attempts, including account lockout or temporary blocking rules.
Forbidden-password dictionary	Ability to maintain a dictionary of prohibited passwords, preventing the use of weak, common, or restricted password values.
Password expiration notifications	Ability to notify users before password expiration according to configured notification rules.

B.3 Applications and data protection (CNF.165)

The solution applies secure input-validation principles in accordance with OWASP Top 10 and OWASP ASVS guidance. Input validation is implemented as a multi-layer control covering the presentation layer, business logic layer, integration/API layer, and data layer. Client-side validation is used to improve user experience and prevent obvious input errors at the user interface level. However, the System does not rely only on client-side validation. All submitted data is also validated on the server side before it is processed, stored, or transmitted to other components. The System validates input data according to defined rules, including data type, mandatory fields, allowed values, length, format, structure, user permissions, and applicable business rules. For structured messages and files, validation may include schema validation, format validation, encoding checks, and rejection of malformed or incomplete requests. At the data layer, database constraints, relationshFBS, uniqueness rules, mandatory fields, and data-format controls are used to preserve data integrity and prevent inconsistent or corrupted data from being stored. Validation errors and rejected requests are logged with relevant technical and business context, including source component, user or system identifier, timestamp, error type, rejected value where appropriate, and corrective action or processing result. These logs support auditability, monitoring, troubleshooting, and security incident investigation. The following evidence demonstrates that the proposed System applies documented secure-development and validation practices based on OWASP Top 10 / OWASP ASVS principles.

Figure 14. B.3-1 – Internal rules for OWASP usages

The secure-development and input-validation rules described above are based on the Supplier's internal technical documentation. The original documentation is maintained in Lithuanian and is available as supporting evidence upon request. The English summary included in this section describes the relevant rules for supplier evaluation purposes:

B.4 Data protection mechanisms (CNF.168)

To ensure the confidentiality of the data, the following measures are applied:

- Encryption;
- Restriction of access rights.

The following measures are used for ensuring integrity:

- Hash;
- MAC;
- Digital signature.

Algorithms for cryptographic operations (hashing, symmetric/asymmetric encryption, MACs, digital signatures) are selected considering all of the following:

- NIST (National Institute of Standards and Technology) "Cryptographic Standards and Guidelines"
- FFBS.
- Local regulatory standards and requirements, if any
- HSM usage:
    - If cryptographic operation is performed by HSM, only HSM supported algorithms, which comply with FFBS 140-3, Security Requirements for Cryptographic Modules | CSRC, are used;
    - Otherwise, priority is being given to the algorithms, which are natively supported by the Oracle database crypto API (dbms_crypto).

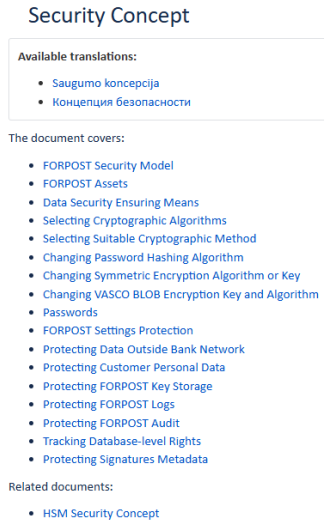
Inside the System, the data integrity and confidentiality are ensured by access control:

- Access to the database tables and API is controlled using secure application roles – only authorized PL/SQL package enables it
- Access to specific entities (customers, accounts, interest schemes etc.) and operations (create customer, view customer, open account, view balance, close account etc.) is controlled using the System user groups and object groups.

Additionally, changes of the data can be tracked using a customizable System audit mechanism.

- View privilege is granted to administrators;
- Depending on the Bank's business processes, view privilege can also be granted to other Bank's employees;
- Only the database schema user (object owner) can directly insert, update, delete operations in the audit table;
- The schema user must be locked;
- The data is recorded into the audit table automatically using the database table triggers on the tracked tables.

confidentiality and integrity of the data, which are transferred outside the Bank's network, is protected by using a secure channel (HTTPS, SSL, VPN). In this case, no additional encryption is required. When transferring the data over insecure channels, the data should be encrypted. Additionally, the data can be digitally signed or MAC-calculated.
Access to highly confidential data is logged for audit and security monitoring purposes. Audit records include, where applicable, the user identity, role, operation performed, object or data category accessed, timestamp, source, result of the operation, and justification for access.
The Supplier maintains an extensive internal **Security Concept** documentation set that describes the security principles, controls, and operational procedures applied in the proposed System. This documentation covers areas such as secure integration, authentication, authorization, encryption, logging, monitoring, auditing, access control, key and certificate management, incident investigation, and operational security. Relevant extracts from this documentation are provided in English as supporting evidence for the security approach proposed for NBM


Security Concept

Available translations:

- Saugumo koncepcija
- Концепция безопасности

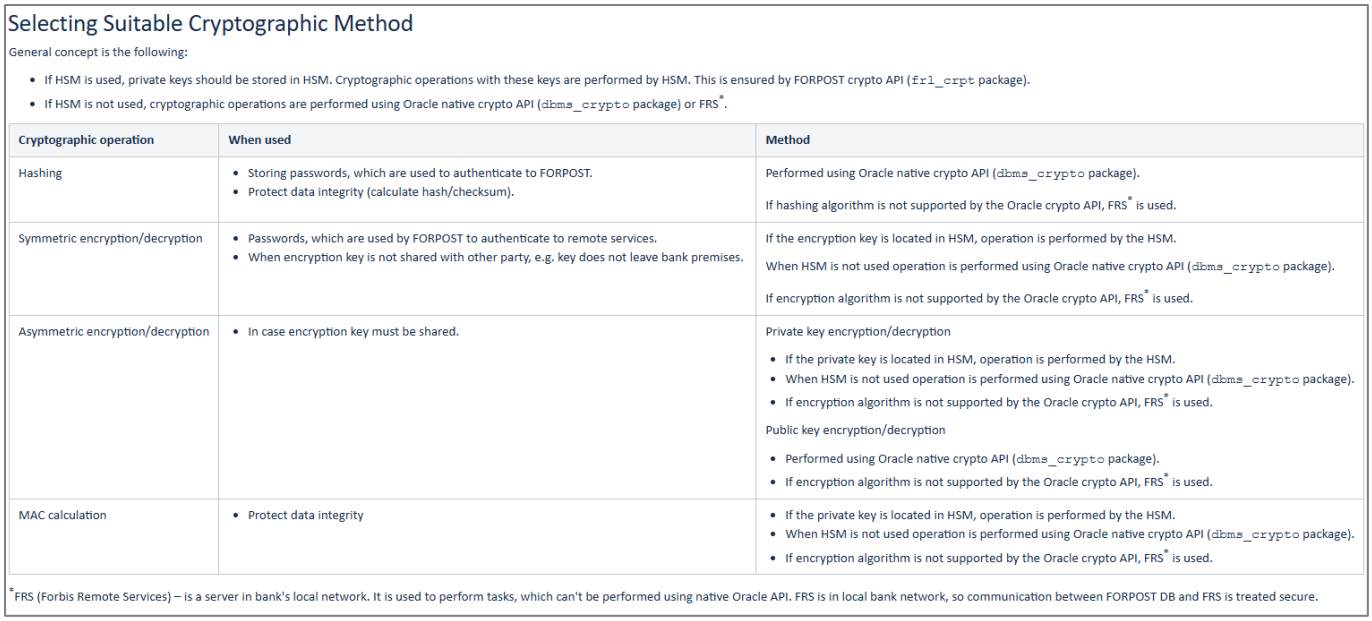
The document covers:

- FORPOST Security Model
- FORPOST Assets
- Data Security Ensuring Means
- Selecting Cryptographic Algorithms
- Selecting Suitable Cryptographic Method
- Changing Password Hashing Algorithm
- Changing Symmetric Encryption Algorithm or Key
- Changing VASCO BLOB Encryption Key and Algorithm
- Passwords
- FORPOST Settings Protection
- Protecting Data Outside Bank Network
- Protecting Customer Personal Data
- Protecting FORPOST Key Storage
- Protecting FORPOST Logs
- Protecting FORPOST Audit
- Tracking Database-level Rights
- Protecting Signatures Metadata

Related documents:

- HSM Security Concept

Figure 15. B.4-1 – "Security concept content"



Selecting Suitable Cryptographic Method

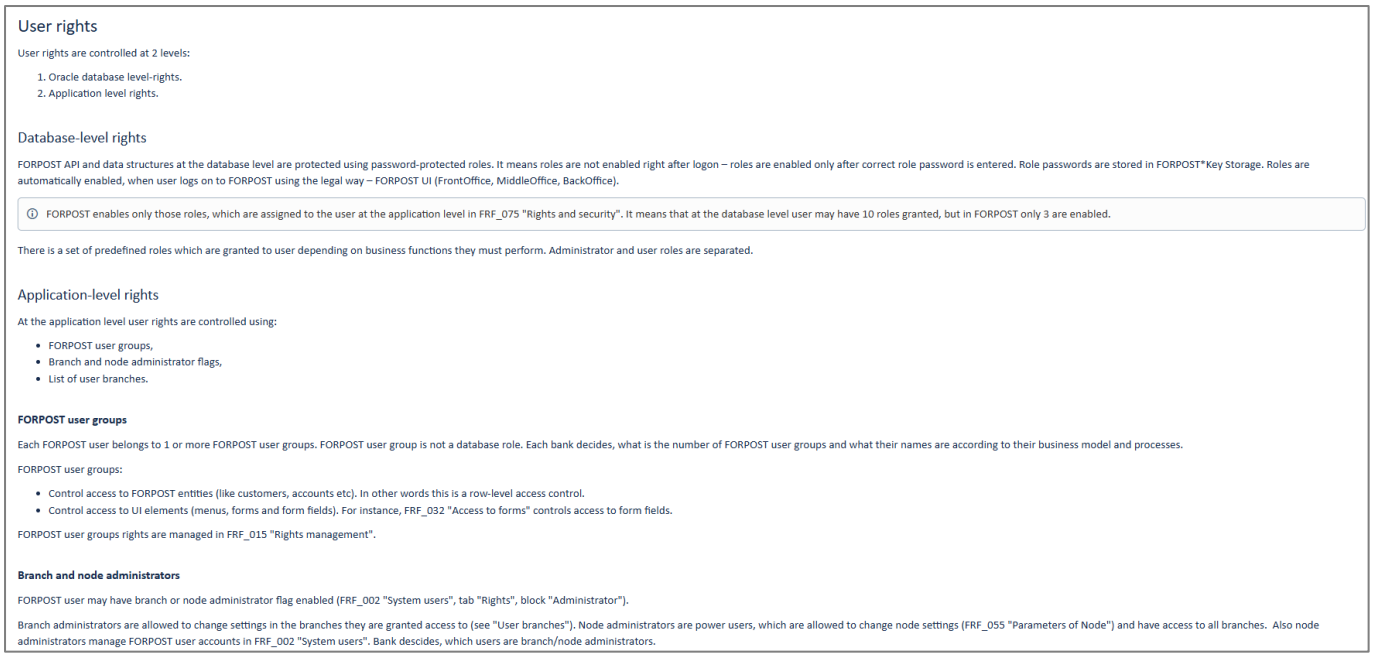
General concept is the following:

- If HSM is used, private keys should be stored in HSM. Cryptographic operations with these keys are performed by HSM. This is ensured by FORPOST crypto API (`frl_crpt` package).
- If HSM is not used, cryptographic operations are performed using Oracle native crypto API (`dbms_crypto` package) or FRS*.

Cryptographic operation	When used	Method
Hashing	• Storing passwords, which are used to authenticate to FORPOST. • Protect data integrity (calculate hash/checksum).	Performed using Oracle native crypto API (`dbms_crypto` package). If hashing algorithm is not supported by the Oracle crypto API, FRS* is used.
Symmetric encryption/decryption	• Passwords, which are used by FORPOST to authenticate to remote services. • When encryption key is not shared with other party, e.g. key does not leave bank premises.	If the encryption key is located in HSM, operation is performed by the HSM. When HSM is not used operation is performed using Oracle native crypto API (`dbms_crypto` package). If encryption algorithm is not supported by the Oracle crypto API, FRS* is used.
Asymmetric encryption/decryption	• In case encryption key must be shared.	Private key encryption/decryption • If the private key is located in HSM, operation is performed by the HSM. • When HSM is not used operation is performed using Oracle native crypto API (`dbms_crypto` package). • If encryption algorithm is not supported by the Oracle crypto API, FRS* is used. Public key encryption/decryption • Performed using Oracle native crypto API (`dbms_crypto` package). • If encryption algorithm is not supported by the Oracle crypto API, FRS* is used.
MAC calculation	• Protect data integrity	• If the private key is located in HSM, operation is performed by the HSM. • When HSM is not used operation is performed using Oracle native crypto API (`dbms_crypto` package). • If encryption algorithm is not supported by the Oracle crypto API, FRS* is used.

*FRS (Forbis Remote Services) – is a server in bank's local network. It is used to perform tasks, which can't be performed using native Oracle API. FRS is in local bank network, so communication between FORPOST DB and FRS is treated secure.

Figure 16. B.4-2 – “Internal Cryptographic Algorithm Selection Guidelines”



User rights

User rights are controlled at 2 levels:

1. Oracle database level-rights.
2. Application level rights.

Database-level rights

FORPOST API and data structures at the database level are protected using password-protected roles. It means roles are not enabled right after logon – roles are enabled only after correct role password is entered. Role passwords are stored in FORPOST*Key Storage. Roles are automatically enabled, when user logs on to FORPOST using the legal way – FORPOST UI (FrontOffice, MiddleOffice, BackOffice).

ⓘ FORPOST enables only those roles, which are assigned to the user at the application level in FRF_075 "Rights and security". It means that at the database level user may have 10 roles granted, but in FORPOST only 3 are enabled.

There is a set of predefined roles which are granted to user depending on business functions they must perform. Administrator and user roles are separated.

Application-level rights

At the application level user rights are controlled using:

- FORPOST user groups,
- Branch and node administrator flags,
- List of user branches.

FORPOST user groups

Each FORPOST user belongs to 1 or more FORPOST user groups. FORPOST user group is not a database role. Each bank decides, what is the number of FORPOST user groups and what their names are according to their business model and processes.

FORPOST user groups:

- Control access to FORPOST entities (like customers, accounts etc). In other words this is a row-level access control.
- Control access to UI elements (menus, forms and form fields). For instance, FRF_032 "Access to forms" controls access to form fields.

FORPOST user groups rights are managed in FRF_015 "Rights management".

Branch and node administrators

FORPOST user may have branch or node administrator flag enabled (FRF_002 "System users", tab "Rights", block "Administrator").

Branch administrators are allowed to change settings in the branches they are granted access to (see "User branches"). Node administrators are power users, which are allowed to change node settings (FRF_055 "Parameters of Node") and have access to all branches. Also node administrators manage FORPOST user accounts in FRF_002 "System users". Bank descides, which users are branch/node administrators.

Figure 17. B.4-3 – “User and applications rights”

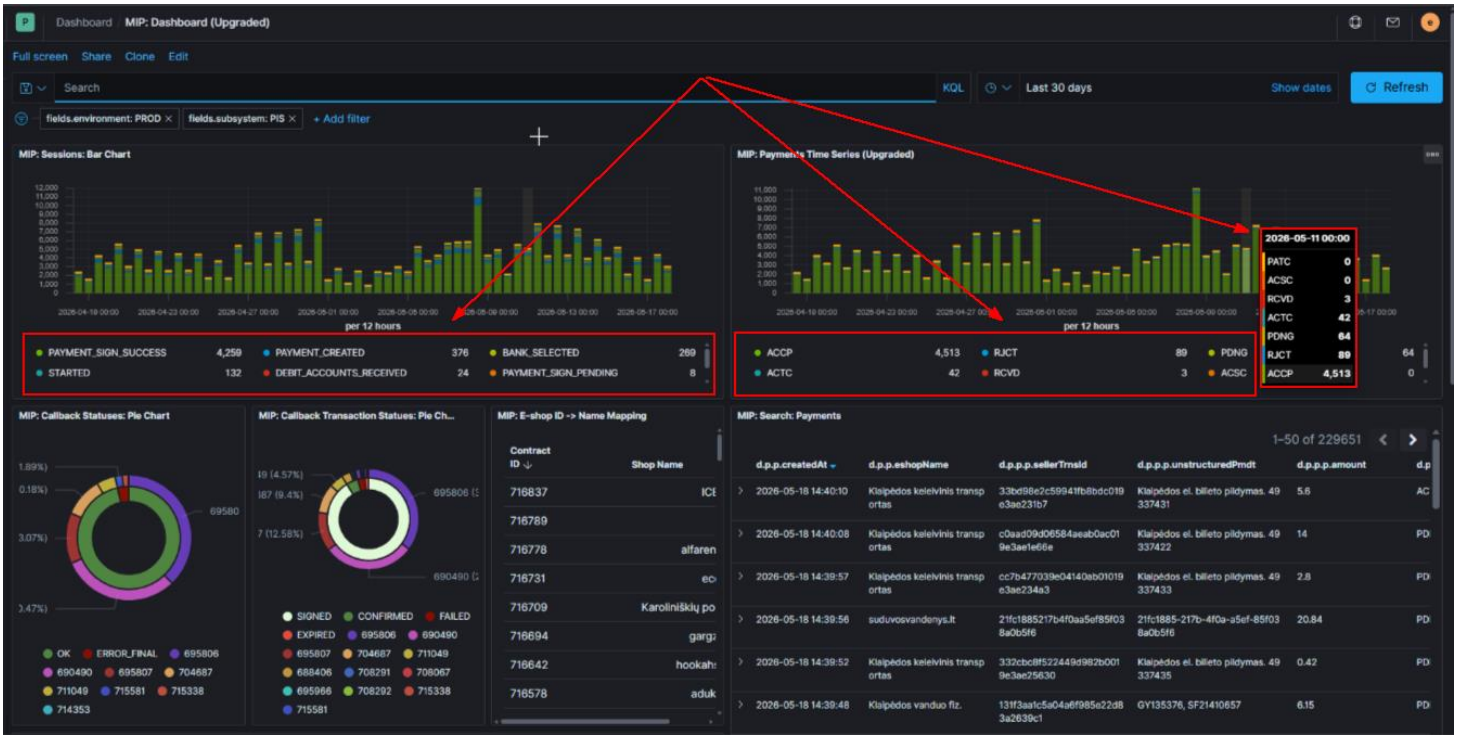
B.5 Audit evidence and event traceability (CNF.135, CNF.178, CNF.180, CNF.181, CNF.183, CNF.188)

The system records user and system activity at business, database and applications levels. Business logs are stored in the database, applications logs are retained in the relevant Java and integration components, and correlation IDs are used to connect end-to-end flows across service calls and database operations. During implementation, the log export and monitoring integration pattern will be agreed with NBM so that relevant logs and security events can be forwarded to the target monitoring or SIEM platform without weakening the existing audit model.

Event category	Audit content	Use
User actions	User identifier, timestamp, action type, function, affected data categories and source context.	Business audit, investigation, accountability and dispute resolution.
Database changes	Insert/update/delete events, old and new values where applicable, table/column context and transaction metadata.	Data integrity investigation and sensitive data history.
Integration transactions	Correlation ID, endpoint, request/response status, routing result, retry status and error details.	Operational monitoring and incident diagnosis.
Reports and exports	Report identifier, user, execution time and parameters where appropriate.	Data access audit and reporting control.
Security events	Authentication events, access failures, privileged actions, policy exceptions and alert triggers.	Security monitoring and incident response.
Trace records	User identifier, timestamp, action type, function, affected data categories and source context.	Investigation.

For back-office users there are users' interfaces (UI) to view logs data with exact time and event evidence.

As example, logs viewer for exchange data with external systems(payments):



Appendix C. Integration principles (CNF.6, CNF.100, CNF.101)

- Integrations are implemented as dedicated microservices or routes with clearly defined ownership, interfaces and operational characteristics.

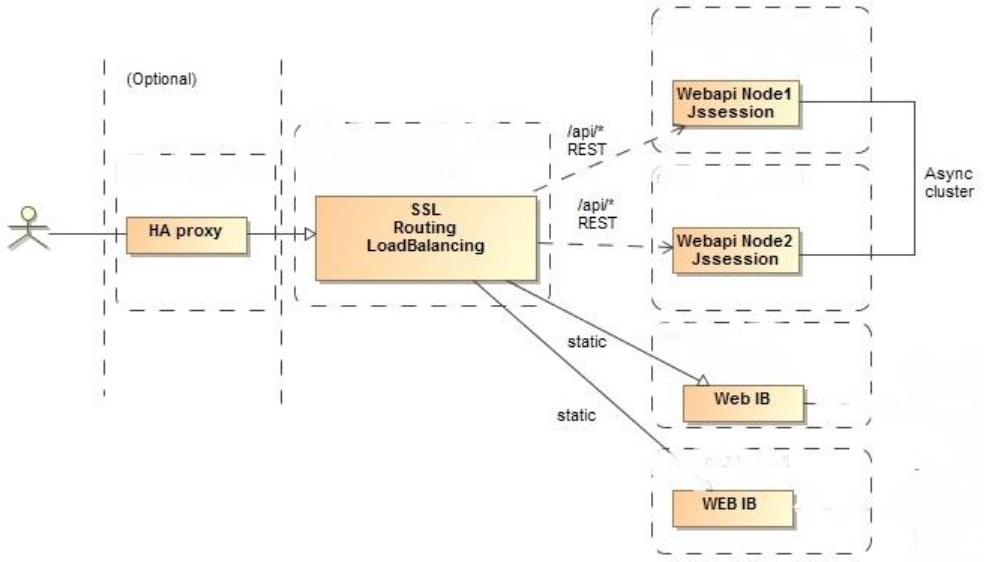
- Integration services use Spring Boot, Apache Camel and Red Hat AMQ Broker where messaging is required.
- Business systems interact with the core banking database only through approved applications APIs and database APIs, not through direct ad hoc database access.
- Interfaces are designed with idempotency, retry, exception handling, correlation IDs and auditability to support reliable banking operations.
- The system supports these protocol/standards:

Protocol/ Standard	How the system implements it	Security and operational controls
ISO 20022	The Integration Platform (ESB) supports processing of ISO 20022 XML-based financial messages where required by NBM integrations. The platform validates messages against the applicable ISO 20022 XSD schemas, maps ISO 20022 structures to internal canonical objects and CBS data models, supports message versioning, and provides configurable transformation rules for inbound and outbound message flows.	Messages are exchanged only through authorized channels. The platform applies schema validation, business-rule validation, full audit logging, correlation IDs, error handling, rejection processing, and message-status tracking. Transport security is provided by the relevant channel, such as HTTPS, SFTP, or SWIFT-related connectivity, as applicable.
SOAP	The system supports SOAP-based web services for integrations requiring formal service contracts. SOAP services are described using WSDL, support XML payloads and XSD validation, and can be exposed or consumed through the Integration Platform (ESB). SOAP endpoints may be used for synchronous request/response integrations and structured system-to-system communication.	SOAP communication is protected through HTTPS/TLS. Where required, WS-Security, message signing, authentication, authorization, IP filtering, request validation, audit logging, timeout control, and error handling are applied. All SOAP requests and responses are logged with technical metadata and correlation IDs.
REST	The system supports REST APIs for internal and external integrations. REST services are exposed through the Integration Platform (ESB) and/or API Gateway, using standard HTTP methods such as GET, POST, PUT and DELETE. API contracts are documented using OpenAPI/Swagger. Payloads is JSON format.	REST APIs are protected by HTTPS/TLS and controlled through authentication and authorization mechanisms such as OAuth2, JWT, API keys, mTLS, or other mechanisms agreed with NBM. The platform supports rate limiting, input validation, versioning, logging, monitoring, correlation IDs.
HTTP/S	HTTP and HTTPS are supported as the standard transport mechanisms for web-service based interfaces, including REST and SOAP. The Integration Platform manages endpoint exposure, routing, request forwarding, response handling, timeout management, retries, and service availability controls.	HTTPS is mandatory for all integrations with NBM other systems or 3rd party services. The platform supports TLS certificate management, trusted certificates, secure cipher suites, mTLS where required, secure headers, network segmentation, firewall rules, access control, request size limits, logging, monitoring, and availability checks.
XML	The system supports XML. XML processing includes parsing, XSD validation, namespace handling, transformation to/from canonical models, and conversion to other formats where required.	XML payloads are validated against agreed schemas before processing. The system applies protection against malformed XML, unauthorized external entities, oversized payloads, and invalid structures.
SFTP	The system supports secure file exchange using SFTP and/or FTPS, according to the protocol required by NBM or the external counterparty. The Integration Platform can send, receive, validate, archive, and process files in agreed formats such as XML, CSV, TXT, or other structured files. It supports scheduled and event-based file processing.	File transfer is protected using SSH keys, certificates, passwords, or other agreed authentication mechanisms. The platform supports encrypted transport, controlled folders, file naming conventions.
SMPTP	The system supports SMTP for sending email notifications, alerts, workflow messages, reports, and operational communications.	SMTP communication is protected using authenticated SMTP and TLS where supported by the mail infrastructure.
SMPP	The system supports SMPP integration for sending SMS notifications through an SMS gateway or telecom provider, where SMS communication is required by business or operational processes.	SMPP connections are established only with authorized SMS gateways using secured network channels and agreed credentials.

Appendix D. Operations / Monitoring / Backup / DR

This appendix describes the operational model, monitoring approach, backup and recovery controls, and disaster recovery design. The proposed solution is intended to operate as a critical banking platform with controlled operational procedures, measurable service levels and auditable recovery mechanisms.

D.1 Service continuity objectives (CNF.195)

Objective	Proposed commitment/design basis
Availability target	Architecture is designed to support a 99.5% service availability objective through redundant service instances, load balancing, database resilience and operational monitoring in all potential places. For example: 
Disaster recovery topology	Active-passive architecture across two geosites, with the primary site operating live services and the DR site prepared for recovery.
RTO	Recovery Time Objective of up to 4 hours for the target critical scope, supported by documented procedures and recovery rehearsals.
RPO	Recovery Point Objective of up to 5 minutes, supported by database replication and controlled operational backup/archivelog handling.
Site switchover	Target site switchover within 1 hour after the decision to invoke DR and completion of prerequisite checks.

D.2 Support and operational governance

Support will be structured into first-line operational monitoring, second-line application/platform support and third-line product/development support. Incident, problem, change and release management processes will be aligned with the agreed service model. Warranty support will include defect correction, operational assistance, root-cause analysis for agreed incidents and maintenance of the solution within the accepted configuration baseline.

Updates may be divided into two categories:

- **System updates** – including the operating system, Oracle database, virtual platform, etc.
- **Suppliers software updates**.

Updates assigned to the first category — system, operating system, and Oracle product updates — are performed in accordance with the manufacturers' documentation.

During a system update, a backup is created, all applications are stopped, and the update is performed.

During operating system updates, usually one of the cluster nodes is stopped. During this time, the other node continues to serve users. Afterwards, a switch-over is performed: the updated node is started, all requests are redirected to it, and the other node is updated. This approach helps ensure uninterrupted node operation.

Recommended schedule for system software updates:

- If there is a serious security vulnerability or a high-priority bug affecting system operation or deployment, the update may be deployed within 1–3 business days.
- For external portal nodes or the DMZ subnet, updates may be deployed every 3–4 months.
- All other components should be updated at least once per year.

Note: Before deploying an update, testing must be performed in the TEST and DEV environments.

D.3 Backup and recovery (CNF.191, CNF.192, CNF.193)

The backup design combines database-native recovery mechanisms with virtual infrastructure backup and component-level configuration backup. Oracle RMAN is used for database backup and recovery, Oracle Active Data Guard supports standby replication, and Flashback capabilities support recovery from logical errors where applicable. Application VM and configuration backups are handled through standard virtual infrastructure backup tools and change-controlled release archives.

Database backups include full/incremental backups, archived logs and validation of recoverability.

Application components are recoverable from versioned deployment packages, container images, configuration backups and infrastructure snapshots where applicable.

Key stores, trust stores, certificates and security configuration are backed up and protected according to approved security procedures.

Recovery procedures include restoration order, dependency checks, validation tests, security checks and business acceptance confirmation.

Backup and recovery tests will be executed before final acceptance and periodically during warranty/support as agreed.

D.4 Disaster recovery operation (CNF.191, CNF.192, CNF.193)

The DR process is organized as a controlled operational procedure. Technical recovery follows a documented sequence covering database role transition, application service activation, DNS/load-balancer routing, integration endpoint validation, security service readiness, business smoke tests and controlled communication to stakeholders. The recovery strategy will be validated through recovery and continuity testing before production acceptance. DR runbooks will define detection, decision-making, failover, service validation, data reconciliation, communication and fallback procedures. Backup and restore procedures will be documented and periodically tested.

Disaster recovery phases

DR phase	Main activities	Expected evidence
Preparation	Maintain standby database, recovery infrastructure, runbooks, access lists and configuration backups.	DR runbook, replication status, backup reports and readiness checklist.

DR phase	Main activities	Expected evidence
Invocation	Declare incident, freeze changes, confirm last data point and approve recovery start.	Incident record and invocation approval.
Technical recovery	Activate standby, start required services, update routing, validate certificates and interfaces.	Recovery logs, service checks and integration validation.
Business validation	Execute banking smoke tests, reporting checks, user login tests and reconciliation checks.	Validation checklist and acceptance sign-off.

Component Backup and Recovery operations

Component	Description	Backup Method	Recovery method
DB	Main centralized database	Hot backup performed at least once per day. Continuous transfer of archived redo logs to a remote destination. Data Guard with at least one standby database.	Switch-over to standby. Recovery from a hot backup using the archived redo log.
FORMS + repository	Back-office administrator console	VM backup performed at least once per day. Snapshot after the deployment of fixes or changes. Additional Forms servers.	Connection to an additional Forms server. VM restore over the existing instance.
HSM	Keys storage, execution of encryption operations.	Additional HSM with the same content. Copies of secure cards. PKI files are uploaded to the system for secure storage.	Connection to an additional HSM. Recovery from MBK cards and loading of keys and certificates.
Applications servers	Applications REST-API services, integration microservices, „Active MQ“.	VM backup performed at least once per day. Snapshot after the deployment of fixes or changes. Additional application servers.	Connection to an additional application server. VM restore over the existing instance.
Authorization server	„Keycloak“ authorization server and dedicated DB.	VM backup performed at least once per day. Snapshot after the deployment of fixes or changes. Additional application servers.	Connection to an additional authorization server. VM restore over the existing instance.

Appendix E. Performance and Test Strategy

This appendix defines the performance strategy for the proposed solution. The objective is to demonstrate that the architecture can meet required banking workloads, remain stable under expected and peak usage.

E.1 Minimum Guaranteed Performance Values Based on Recommended Technological Platform (CNF.87)

Performance indicator	Minimum guaranteed value	Measurement condition
Concurrent active user sessions	Minimum 100 concurrent active users	Normal banking-day activity, based on recommended infrastructure sizing.
Daily transaction processing capacity	Minimum 2,000,000 transaction/accounting records per business day	Based on OLTP processing capacity of the recommended platform; to be validated by load test.
SWIFT / ISO 20022 message-processing capacity	Minimum 10,000 messages per business day	Inbound/outbound financial message processing, excluding external network delays.
Payment document processing capacity	Minimum 200,000 payment documents per business day	Batch or real-time processing depending on interface type.
Day-end closing processing	Completed within 2 hours	Normal business-day volume.
Month-end closing processing	Completed within 3 hours	Normal monthly closing workload.
Year-end closing processing	Completed within 4 hours	Normal annual closing workload.
Standard operational report generation	≤ 10 seconds in at least 90% of cases	Standard predefined reports on operational/indexed data.
Complex report generation	Executed asynchronously	User may continue working; result is made available after background processing.
Historical data access	Current-data transaction performance remains within guaranteed values with 5 years of retained data	Based on archiving, partitioning, indexing, and optimized access paths.
Integration message processing	Minimum 50 messages/second peak for short intervals; minimum 20 messages/second sustained	ESB/API/integration platform processing, excluding external counterparty delay.
System availability	≥ 99.7% per monthly reporting period	Production environment, excluding agreed maintenance windows if contractually accepted.

The above values are guaranteed for the recommended technological platform and final approved sizing. Compliance will be validated through performance, load, stress, and endurance testing agreed with NBM before production go-live. External network latency and third-party system response time are excluded from applications side measurements.

The specified times for the Day/Month/Year closing processes must be apply only if NBM uses solely the banking products developed by the supplier participant and the end-of-day closing processes described by the tender participant. If the system functionality is expanded at NBM's initiative without the tender participant's consent, the defined times shall not apply.

E.2 Response Time for Standard Online Transactions (CNF.88, CNF.89)

Technical solution	How it supports the requirement
Optimized transaction processing	Business operations are implemented with efficient service logic and controlled transaction boundaries.
Short-lived database transactions	Transactions are kept short to reduce locking, waiting time, and rollback impact.
Database indexing and query optimization	Frequently used queries for balances, accounts, payments, and recent transactions use optimized execution plans and indexes.
Connection pooling	Database and service connections are reused to reduce connection overhead and improve response time.
Caching of reference / non-sensitive data	Stable reference data is cached where applicable to reduce repeated database calls.
Scalable application deployment	Application services can run in multiple instances and scale horizontally according to workload.
Workload separation	Online user transactions are separated from batch jobs, reports, and background processing.
Asynchronous processing	Long-running processes such as complex reports, EOD/EOM/EOY, imports, exports, and recalculations are executed in background.
Integration layer controls	External service calls are managed through API / Integration Platform controls, including timeout, retry, logging, and correlation ID.
Monitoring and diagnostics	Response time, throughput, errors, database load, and resource usage are monitored to detect and resolve performance issues.
Performance testing	Compliance will be validated through agreed performance tests and documented in the Performance Test Report.

Validation item	Acceptance criterion
Standard GUI transactions	At least 95% complete within 2 seconds
Standard external service/API transactions	At least 95% complete within 2 seconds, excluding third-party/network delay
Test condition	Normal operating load, excluding EOD/EOM/EOY and complex reports
Evidence	Performance Test Plan, Performance Test Report, monitoring dashboard

Appendix F. Source Code / Escrow / Development Governance

This appendix describes the governance model for source code, escrow, development controls, build/release management and change assurance. The objective is to protect NBM's continuity interests while maintaining a controlled and auditable delivery process.

F.1 Secure development lifecycle (CNF.198)

The Supplier maintains internal secure system development documentation titled **"Secure System Development Policy and Principles."** The documentation defines mandatory rules for secure and maintainable development of internal IT systems, integration solutions, and systems developed by third-party providers. The policy is aligned with **LST EN ISO/IEC 27001:2022** requirements and applies across the full software development lifecycle. The documentation establishes that all developed systems must be based on security principles and that information security and privacy protection must be included in every stage of system development. The core principles include protection of **confidentiality, integrity, and availability**, secure processing, transmission and storage of information, regular review of secure-development principles,

and evaluation of security/privacy risks when new technologies or architectural changes are introduced. The policy requires that developers, architects, testers, and other relevant personnel have appropriate information-security competence.

Architects and developers must complete secure-development and system-security training, continuously improve their knowledge, review security testing results, identify new risks and attack vectors, and update internal security practices.

During requirements definition and design, the documentation requires that confidentiality, integrity, availability, and privacy needs are identified and included in requirements specifications. System design must apply security-by-design principles, attack-surface analysis, least-privilege principles, minimization of personal-data processing, separation of user and administrator roles, modular/layered architecture, auditability, and controlled use of trusted components. The policy also requires assessment of sensitive data when creating new database structures or adding new fields.

The documentation defines secure environment preparation rules. When test environments are prepared using production copies, anonymization scenarios must be executed, and anonymization reports must be produced and reviewed to confirm correctness.

For software development, the documentation requires compliance with internal programming practices, general good coding practices, and system-security principles. Source code must be stored in a source-code repository. Third-party APIs, libraries, and components may be used only after security and suitability assessment, approval by responsible architecture personnel, and vulnerability checks. Periodic scanning of third-party libraries for newly identified vulnerabilities is required, and vulnerable components must be updated or replaced when needed.

The documentation also defines verification and quality-control rules. Code review must be performed by a system architect and/or senior developer, and verification must be performed by a person other than the developer. Solutions must be tested using defined manual and automated testing methods. Common errors and vulnerabilities must be identified and appropriate protections implemented.

Before and after release of new software versions, security testing must be performed. Initial security testing is carried out by the Supplier’s specialists, while detailed testing may be performed by the client in an environment close to production or through an independent audit. Security issues identified during testing must be corrected immediately, and security fixes must be provided to clients without delay.

This internal documentation supports the maintainability and quality of delivered source code by defining secure-development governance, coding and architectural principles, modular/layered design, source-code control, code review, third-party dependency control, vulnerability management, manual and automated testing, security testing, release verification, and prompt remediation of detected security issues.

The original internal documentation is maintained in Lithuanian; therefore, it is not attached in full to the tender documentation, but the relevant English summary and selected evidence extracts are provided for evaluation purposes.

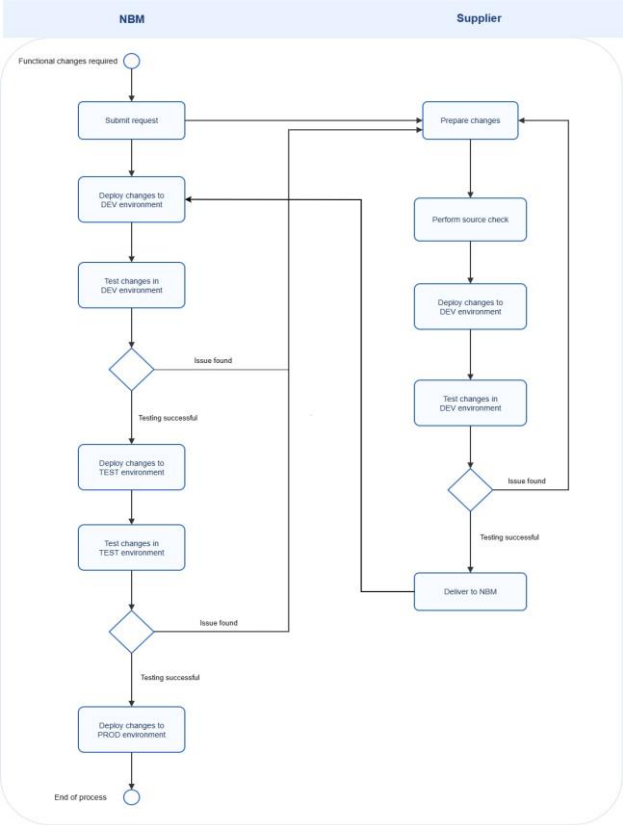
In addition to the secure-development principles described above, the delivered source code follows maintainable software development practices, including clear and consistent project/package/module structure, meaningful and self-explanatory naming of variables, functions, classes, procedures, and database objects, and inline comments for complex business logic, algorithms, integrations, and non-obvious implementation decisions. The source code is designed according to modular and layered

architecture principles to support future maintenance, scalability, and modification. Before delivery, the source code is subject to quality assurance checks, including code review, static code analysis where applicable, unit testing, and verification against the agreed coding guidelines and quality requirements, including ISO/IEC 25010-aligned maintainability principles where applicable.

F.2 Configuration and changes management (CNF.198)

Configuration is separated from source code where practical and controlled through environment-specific parameter sets, secrets management, certificates, trust stores and deployment descriptors. The changes (source code or configuration or both) are implemented and delivered in accordance with the defined scheme. The same process steps are applied to both critical and non-critical changes.

Figure 18. F.3-1 – ”Changes management”



More detailed information on Forbis Quality Assurance and Change Management is provided in the document Forbis_Change_Management.dox

No.	Step	Responsibility
1.	Request preparation	NBM
2.	Change implementation and deployment	Supplier
3.	Deployment of the change to the DEV environment	Supplier
4.	Testing of the changes in the DEV environment	Supplier
5.	Evaluation of testing results	Supplier
6.	Deployment of the changes to the DEV environment	Supplier
7.	Testing of the changes in the DEV environment	Supplier
8.	Evaluation of testing results	Supplier
9.	Deployment of the changes to the TEST environment	Supplier
10.	Testing of the changes in the TEST environment	NBM, Supplier
11.	Evaluation of testing results	NBM, Supplier
12.	Deployment of the changes to the PROD environment	NBM, Supplier

F.3 Future Development Governance and Third-Party Extensibility (CNF.103, CNF.180)

NBM and third parties authorized by NBM may develop additional components, integrations, reports, workflows, and extensions using the documented methodology and technical framework provided by the Supplier. This framework includes development standards, naming and coding conventions, architectural

principles, API and integration specifications, security rules, testing procedures, deployment rules, versioning approach, and quality-control requirements.

The proposed solution does not impose commercial or technical restrictions that would prevent NBM from carrying out internal developments or outsourcing such developments to third parties, provided that the agreed methodology and technical framework are followed. The mandatory involvement of the Supplier's personnel is not required for future developments, except where NBM explicitly requests support or specialized assistance.

The application supports multiple extensibility options to enable future functional and technical enhancements, including:

- extensibility of banking product functionality to add new functions to existing products or create new products;
- extensibility of payment-processing functionality to add new functions to existing payment processes or create new ones;
- extensibility of S2S platform services for communication with external parties;
- extensibility of the data-exchange subsystem to add new import/export channels;
- extensibility of the notification subsystem;
- extensibility of day/month/year closing functionality;
- extensibility of asynchronous processes;
- possibility to create new reports;
- possibility to create new user tasks.

All future developments must comply with the approved architecture, security model, integration rules, quality-control procedures, and release-management process in order to ensure compatibility, interoperability, maintainability, and upgrade safety.

F.4 Source code Escrow and Verification Procedure (CNF.196, CNF.197)

The Supplier supports source-code escrow as a business-continuity and vendor-risk mitigation mechanism. The purpose of the escrow arrangement is to ensure that NBM can obtain the source code and related technical materials if the software supplier becomes unable to maintain or support the delivered System, including cases such as liquidation, bankruptcy, reorganization, or other release events agreed by the Parties. The escrow agreement will be concluded with a reputable escrow agent mutually agreed with NBM. The escrow arrangement may be activated at NBM's discretion after final acceptance of the solution. Unless otherwise agreed by the Parties, the escrow package will be submitted within 30 working days and maintained for a minimum period of five years.

The escrow package will include the materials necessary to enable independent maintenance and further development of the System, including:

Escrow package item	Description
Source code	Source code of the application components included in the proposed System.
Build instructions	Instructions required to compile, package, and verify the software.
Deployment instructions	Procedures for installing and deploying the System in target environments.
Database scripts	Database schema, migration scripts, stored procedures, and required database objects.

Escrow package item	Description
Configuration templates	Environment configuration files, parameters, and deployment templates.
Dependency list	Third-party libraries, frameworks, versions, runtime dependencies, and licensing references.
Interface documentation	API specifications, integration contracts, message formats, and communication rules.
Technical documentation	Architecture, development, administration, operation, and maintenance documentation.
Verification materials	Instructions and test evidence needed to confirm that the deposited source code corresponds to the delivered System.

For third-party components, the escrow package will include the source code, dependency materials, documentation, or access instructions to the extent permitted by the applicable licensing terms. Where direct deposit of third-party source code is not legally possible, the supplier will provide the necessary dependency list, version information, licensing references, configuration details, and replacement or acquisition instructions required for independent maintenance.

The supplier also supports escrow verification exercises. Such exercises are used to confirm that the deposited materials are complete, correspond to the delivered solution version, and can be used to build, maintain, or further develop the System. The Supplier has experience with escrow agreements and escrow verification activities performed for other clients; related evidence can be provided subject to confidentiality restrictions.

F.5 Source code Package Integrity, Authenticity and Secure Transfer (CNF.199)

The supplier will protect all source-code delivery packages using controls that ensure authenticity, integrity, confidentiality, and traceability during preparation, transmission, storage, and verification. All files containing source code will be delivered as a controlled source-code package. Each package will be identified by a unique package name, version, delivery date, and release reference. The package will include source code, build instructions, deployment instructions, dependencies, database scripts, configuration templates, and other agreed technical documentation. To confirm the origin and integrity of the delivered source code, the package will be digitally signed by the Supplier. The digital signature will allow NBM to verify that the package was prepared by the supplier and that the contents have not been modified after signing. Where applicable, the digital signature will be timestamped to provide evidence of the signing time and to support auditability and legal traceability. Source-code packages will be encrypted during transmission and protected during storage to prevent unauthorized access. Secure transfer channels and/or encrypted archives will be used according to the agreed source-code delivery or escrow procedure. Access to source-code packages will be limited to authorized personnel only.

The supplier will provide verification instructions describing how to validate the source-code package. The verification process will include checking the digital signature, timestamp, package integrity, package version, and completeness of the delivered materials. For audit and legal traceability purposes, the source-code delivery and verification process will be documented. The records may include package identifier, version, checksum or signature reference, timestamp, delivery date, sender, recipient, storage location, verification result, and any detected exceptions.

Appendix G. Partner / Provider Evidence

This appendix defines the evidence model for the suppliers, product providers and support chain associated with the proposed solution. It is intended to demonstrate that the proposed architecture is not only technically feasible, but also supportable during implementation, warranty and subsequent operation.

G.1 Provider responsibility model

Area	Responsible provider role	Evidence / assurance expected
Core banking application	Application supplier and implementation team.	Product description, implementation methodology, architecture documentation, references and warranty/support commitment.
Database platform	Oracle licensing and database administration support chain.	License entitlement, version compatibility, support status and sizing confirmation.
Container platform and middleware	Red Hat / OpenShift / JBoss EAP / AMQ support chain and implementation team. Oracle Cloud Infrastructure (OCI) / Oracle Middleware support chain and implementation team. Rancher RKE2 / Kubernetes platform support chain and implementation team.	Subscription model, supported versions, deployment topology and operational support responsibilities.
Reporting platform	JasperReports and Microsoft Office reporting service support chain.	License/subscription confirmation, deployment model and report customization approach.
Security components	Keycloak, WAF, certificates, key stores/trust stores, optional HSM and anti-malware service owners.	Security design, certificate/key management process, hardening guidance and operational ownership.
Infrastructure	NBM infrastructure team with supplier architectural guidance.	Network/firewall rules, backup responsibilities and monitoring integration.
Escrow/source continuity	Supplier and agreed escrow provider where applicable.	Escrow arrangement or source code delivery commitment, deposit content and update procedure.

G.2 Evidence package structure

The tender and implementation evidence package will be maintained as a controlled set of documents and confirmations. The package will allow the tender commission and technical evaluators to verify the proposed architecture, support arrangements, implementation readiness and continuity commitments.

- Product and architecture descriptions demonstrating conformity with the non-functional requirement topics.
- Technology stack and licensing assumptions for production and non-production environments.
- Implementation methodology, project governance, acceptance gates and quality management approach.
- Provider support chain, warranty model and escalation process.
- Security, backup, DR, monitoring and operational procedures.
- Source code delivery or escrow arrangements, subject to contract and third-party licensing constraints.
- Training, knowledge transfer and handover evidence for administrators, operators and business users.

G.3 Support assurance and warranty readiness

The support model will identify the responsibilities of the application supplier, platform providers, infrastructure team and NBM operational stakeholders. The model will define incident escalation, severity

classification, communication paths, defect correction, patch coordination, security advisory handling and release planning. During the warranty period, accepted incidents and defects will be handled through a controlled support process, and recurrent or critical issues will be subject to root-cause analysis and corrective action tracking.

G.4 Licensing and subscription governance

License and subscription quantities will be confirmed based on the final deployment topology, user volumes, transaction throughput, resilience requirements and vendor licensing rules. Perpetual or subscription licensing assumptions will be documented for each relevant component, with the objective of ensuring that the delivered system can be legally operated, supported and maintained without hidden dependencies.