Руководство разработчика BIV Platform

Введение

Платформа BIV — это набор инфраструктурных библиотек и сервисов для создания и развития информационных систем для финансовых организаций в Микросервисной Архитектуре.

Микросервис – java-приложение, разработанное с использованием фреймворка quarkus, как правило с rest-api (это определение микросервиса касается разработки на платформе BIV и не является общим!).

Фронтенд микросервиса – микрофронтенд – SPA-приложение, фактически это пользовательский интерфейс микросервиса. В некоторых случаях у микросервиса может не быть микрофронтенда.

Информационная система – набор микросервисов, баз данных, других хранилищ, очередей сообщений и других сервисов, предназначенных для автоматизации бизнеса.

Руководства и правила

Разработка бэкенда приложений на платформе BIV

Базовым фреймворком платформы является Quarkus.

Правила разработки см. «Руководство разработчика по использованию фреймворка Quarkus».

Работа данными (бэкенд)

Используемая СУБД определяется в рамках конкретного проекта. Рекомендуемые СУБД: PostgreSQL, MongoDB.

Расширение для Quarkus entity-info.
Расширяет описание модели данных (сущностей) в ORM информацией на естественном языке в терминах предметной области. Расширенное описание предметной области позволяет бизнес-пользователю самостоятельно конфигурировать систему. На базе данного расширения строятся все остальные библиотеки, расширения и сервисы.
См. «Руководство разработчика entity-info»

Библиотека distributed-entity-info — расширение entity-info.
Собирает распределённое описание модели данных по нескольким микросервисам. Позволяет составлять запросы к данным, располагающимся в разных микросервисах,

осуществляет динамическое связывание данных из разных микросервисов.

Библиотека service-discovery-api – библиотека для унифицированного взаимодействия с Consul. Требуется для работы distributed-entity-info.

Библиотека entity-object-template для универсального создания объектов по шаблону. Позволяет настроить создание предзаполненного объекта сущности из расширенной модели данных, например, договора страхования согласно настройкам страхового продукта. Содержимое шаблона настраивается бизнес-пользователем без перекомпиляции исходного кода системы. Предоставляет API для создания предзаполненного объекта сущности.
См. «Руководство разработчика entity-object-template».

Библиотека refbook для описания и хранения произвольных справочных данных. Позволяет эффективно настраивать НСИ без перекомпиляции исходного кода системы. Дополняет расширенную модель данных. Определят микросервис, являющийся владельцем справочника, и микросервисы-потребители данных справочника. Встроенный механизм репликации данных позволяет синхронизировать содержимое НСИ между микросервисами. Предоставляет API для использования данных НСИ в информационной системе.
См. «Руководство разработчика refbook».

Библиотека state-machine — машина состояний, основанная на расширенной модели данных.
«Навешивается» на сущность из расширенной модели данных. Позволяет описать граф переходов состояний и триггеры перехода. Осуществляет перевод объекта сущности между состояниями согласно графу, выполнение триггеров состояния.
См. «Руководство разработчика state-machine».

Библиотека entity-journal для настраиваемого отображения журналов.
Позволяет пользователю настраивать представление табличных данных на основании расширенной распределённой модели данных. В рамках представления определяются состав колонок, формат отображаемых данных, их сортировка. Дополнительно определяются критерии выборки данных, которые должен указать в ходе работы пользователь. Пользователь может создать несколько представлений и переключаться между ними. Отображение данных осуществляется с учётом прав пользователя.
См. «Руководство разработчика entity-journal».

Работа с идентификаторами

Проблематика: при разработке API возникает проблема передачи внутренних идентификаторов объектов системы.

Правильно: /account/dywe93mq

Не правильно: /account/1234 – ИД объекта в системе

Требования к ИД при передаче во вне: ИД должен быть таким, чтобы нельзя было перебирать объекты системы найдя последовательность, например, целые числа.

Хранение идентификаторов объектов в БД микросервисов: ИД объектов и ссылки на них в БД микросервисов должны быть целочисленными.

Взаимодействие микросервисов по внутреннему api происходит без шифрования ИД.

Таким образом, API сервиса во вне нельзя выставлять на прямую, всегда нужно прикрывать gate’ом.

API gate точка в которой должно быть сделано:

- Открытие API для внешнего доступа
- Шифрование и дешифрование ИД
- Мост между внешней моделью аутентификации (oauth2, saml, jdbc) и внутренней аутентификацией (JWT)

Для шифрования и дешифрования ИД необходимо использовать библиотеку платформы crypto-service.

Разработка API микросервиса

Для реализации API необходимо использовать библиотеку rest-common. Библиотека обеспечивает поддержку спецификации запроса и ответа для rest-api микросервисов, обработку и пробрасывание исключительных ситуаций.

Правила разработки API бэкенда описаны в документе «Правила проектирования API».

Разработка фронтенда приложений на платформе BIV

Базовым фреймворком для разработки фронтенда платформы является Angular.

Правила разработки см. «Руководство разработчика по использованию фреймворка Angular».

Разработка фронтенда должна вестись с использованием микрофронтендов.

Взаимодействие backend-frontend. Микрофронтед и микросервис

Микрофронтед является прямым продолжением идеи микросервисной архитектуры для фронтенд-разработки. Таким образом, по аналогии с микросервисом, микрофронтенд – это небольшая, логически отделенная часть приложения, а точнее, в контексте разработки frontend, часть пользовательского интерфейса вашего приложения. Часть, которая в первую очередь не зависит от остальных частей системы с точки зрения ее развертывания. Кроме того, Микрофронтед должен инкапсулировать некоторую часть пользовательского интерфейса бизнес-логики приложения и должен принадлежать исключительно команде, которая отвечает за эту часть приложения от начала до конца.

Правила взаимодействия микрофронтедов и микросервисов

1. Микрофронтенд может только с тем микросервисом, которому он принадлежит (это «свой» микросервис).
   Запрещается вызывать напрямую из микрофронтенда «чужой» микросервис. Если есть необходимость в вызове функционала «чужого» микросервиса, то необходимо «проксировать» вызов «чужого» микросервиса через «свой» микросервис. Т.е. в «своём» микросервисе сделать endpoint, через который будет осуществляться вызов другого микросервиса.

2. Часть пользовательского интерфейса микрофронтенда, который может использоваться в других микрофронтендах, необходимо выносить в отдельные компоненты.
   Данные компоненты должны также вызывать только «свой» микросервис.
3. Общие компоненты должны иметь чёткое версионирование.
4. На текущий момент без явной необходимости не применять шаблон BFF.
   Под явной необходимостью понимается, например, разные типы клиентских приложений (десктоп-веб, мобильный клиент).

Проблемы подхода с выделением компонент

С ростом количества вызов «чужого» функционала растёт потребность выносить всё больше вызываемого интерфейсного функционала в отдельные компоненты.

Это приводит к:

- Необходимости чётко отслеживать версии и совместимость версий при сборке стенда.
  Может получиться так, что для разных микрофронтендов могут потребоваться разные версии одной и той же компоненты.
- Кроме того, поскольку версия компоненты зависит ещё и от микросервиса, то цепочка зависимостей удлиняется. Это своего рода «монолитизация» приложения и повышение зависимости команд друг от друга.

Пока будем считать, что объём одновременно разрабатываемого нами функционала и количество команд позволяют решить эти проблемы аккуратным выстраиванием процесса разработки и сборки.

См. «Руководство разработчика по использованию концепции микрофронтенд».

Разработка web-интерфейса

Библиотека локализации веб-приложений i18n.
Позволяет осуществлять перевод приложения на несколько языков.
См. «Руководство разработчика по локализации веб-приложений».

Микросервис навигации menu.
Осуществляет компоновку веб-приложения, состоящего из микросервисов и микрофронтендов.
См. «Руководство разработчика сервис навигации menu».

Аутентификация и авторизация в приложениях

Для аутентификации и авторизации в микроскервисах используется IDP Keycloak.

См. документацию на сайте проекта https://www.keycloak.org.

В рамках платформы для keycloak разработаны:

- расширение «Профильные права»
  См. «Руководство разработчика Профильные права»
- Расширение «Интеграция с ЕСИА» для аутентификации в информационной

системе через Единую систему идентификации и аутентификации .
См. «Руководство разработчика ЕСИА».

Инфраструктурные сервисы

Сервис формирования печатных форм по шаблону.

- Шаблоны печатных форм создаются в офисном пакете с помощью специальных управляющих конструкций (разметки).
- Поддерживаются шаблоны текстовых документов и электронных таблиц.
- Поддерживается формат OpenOffice/LibreOffice (.ODS и .ODT) и MS Office (.DOCX и .XLSX).
- Позволяет формировать печатную форму с изменяемой структурой (переменное количество разделов, скрываемые / динамические параграфы и т. д.).
- Включает в себя блок преобразования полученных документов в формат PDF на основе LibreOffice. В данном блоке преобразования осуществляется запуск нескольких экземпляров LibreOffice в режиме сервиса, контроль работоспособности этих экземпляров, их перезапуск при необходимости.

См. «Руководство пользователя МСП», «Руководство разработчика libreferma».

Сервис расчёта «Тарифный калькулятор».

- Позволяет бизнес-пользователю самостоятельно в графическом виде описать алгоритм расчёта в том числе с использованием тарифных таблиц.
- Выполняет вычисления по запросу других микросервисов, например при расчёте страховой премии по договору страхования.

Адаптер СМЭВ (Система межведомственного электронного взаимодействия).

- Позволяет подключиться к СМЭВ и обмениваться данными.