На предприятиях, как правило, существует большое разнообразие программ, решающих самые разные задачи: от тривиального бухгалтерского и фискального учета до сложных аналитических комплексов и систем электронного документооборота. Данные, генерируемые этими программами, будут нужны ИНКе, а данные генерируемые ИНКой - понадобятся внешним программам для решения тех или иных бизнес-задач.

Отсюда возникает необходимость в инструменте, обеспечивающем полноценный обмен данными, их полноту и непротиворечивость, а также актуальность, исключающем ошибки ручного ввода, размножение мест хранения одних и тех же данных. Как опцию или бонус, можно рассматривать режим работы этого инструмента, как обмен данными между любыми, подключенными системами, а не просто "внешняя система" ↔ ИНКА. Таким образом, модуль может выступать общим интеграционным решением всего предприятия, а не решать просто локальную задачу доставки данных в ИНКА и добычи их из нее.

Описание функционального модуля Интеграция

Модуль предназначен для дополнения ядра платформы ИНКА средствами интеграции с внешними системами. В основе принципов построения архитектуры модуля лежит микросервисный подход, что позволяет говорить о гибкости и высокой адаптивности данного модуля.

Цели создания модуля:

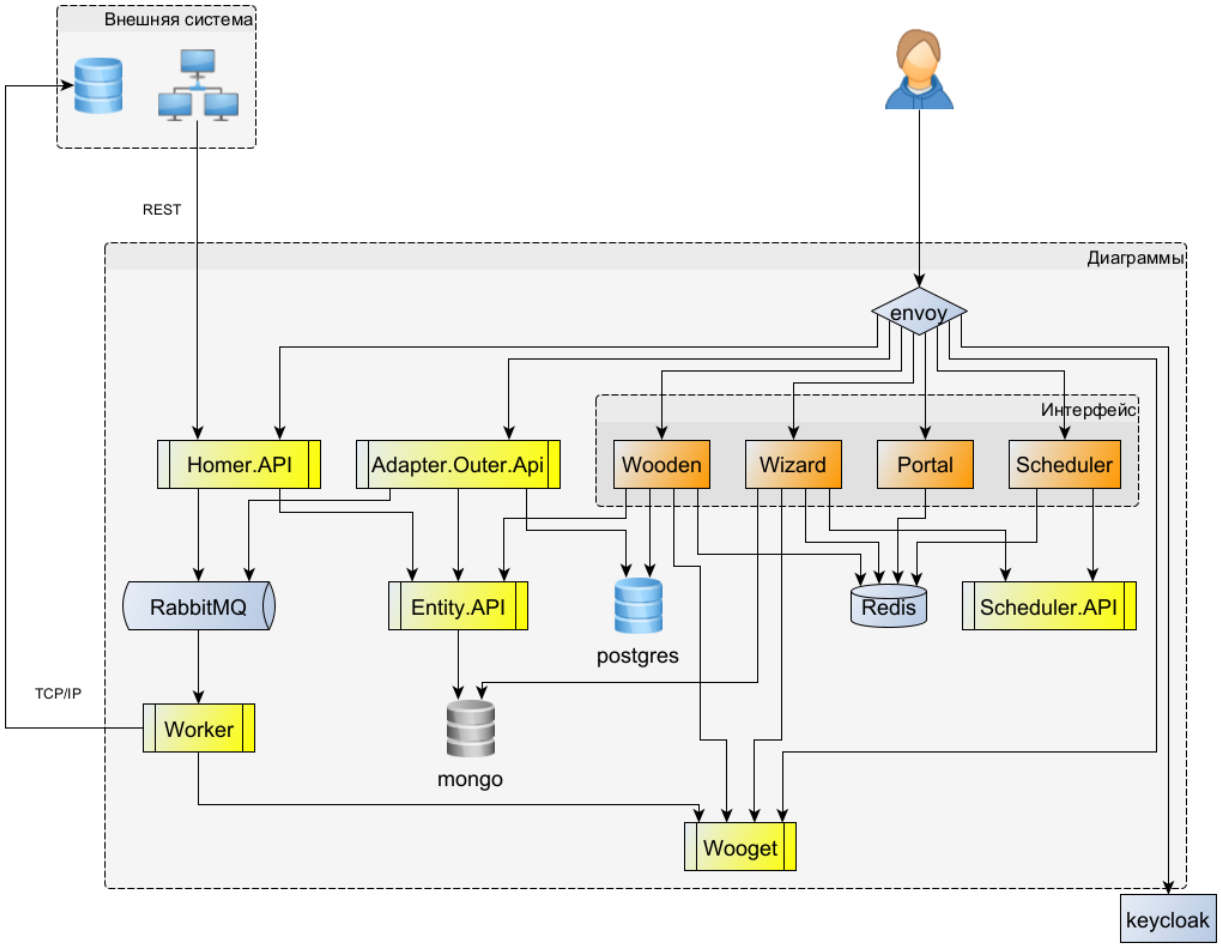
- Обеспечение доступа к данным внешних систем
- Обеспечение внешних систем данными ИНКА
- Обеспечение внешних систем данными друг друга
- Преобразование формата получаемых данных к внутреннему представлению ИНКА и их абстракция от системы-отправителя
- Преобразование исходящих данных в формат системы-получателя

Функциональный модуль позволяет выполнить следующие задачи:

1. Легкую стыковку системы ИНКА с внешними системами (и внешних систем между собой)
2. Оперативное получение данных извне, преобразование их к внутреннему виду системы, доставку в ядро и/или функциональным модулям
3. Снабжение внешних систем требуемыми ими данными в нужном им формате.
4. Декларативное описание алгоритмов загрузки/выгрузки данных, а также их преобразований
5. Механизм "гарантированной" доставки данных, основанный на обмене квитанциями/флагами
6. Расширяемость. Легкую, относительно, возможность доработать модуль под практически любые формы обмена данными
7. Универсальность. Причин для обхода модуля для решения каких-то специфических интеграционных задач должно быть как можно меньше.

8. Безопасность.
9. Транзакционость.

Схема взаимодействия сервисов



Пользовательский интерфейс (UI)

1. Wooden (Диаграммы) - интерфейс для работы с диаграммами.
2. Wizard (Мастер создания сценариев) - интерфейс для быстрого и легкого формирования сценариев интеграции в виде диаграмм.
3. Scheduler (Планировщик) - интерфейс для работы с сервисом Планировщика.
4. Portal (Портал) - интерфейс единого доступа к интерфейсам "Диаграммы", "Планировщик" и "Мастер сценариев".

API

1. Wooget (Менеджер расширений) - сервис для управления расширениями: позволяет хранить расширения и предоставлять к ним доступ через HTTP.
2. Homer (Тут нужно придумать нормальное русское название) - сервис для обеспечения интеграции в разрезе приема/передачи посредством стандартных протоколов (REST, SOAP).
3. Worker (Обработчик заданий) - сервис для выполнения заданий по сборке диаграмм: получение задания из очереди, непосредственно выполнение и формирование ответа на задание.
4. Entity (Сервис сущностей) - сервис для хранения общих сущностей, которые необходимы для работы других сервисов.
5. Adapter.Outer (Внешний адаптер) - сервис для взаимодействия с ИНКА: используется для вызова функций/событий в ИНКА, а также для получения сущностей из нее.
6. Scheduler (Планировщик) - сервис для работы с механизмом запуска задач по расписанию на основе Quartz.NET.

Сторонние сервисы

1. RabbitMQ - программный брокер сообщений на основе стандарта AMQP (используется для организации обмена сообщениями, а также для формирования очереди на расчет/выполнение диаграмм).
2. Envoy - высокопроизводительный распределённый прокси-сервер, спроектированный как для отдельных сервисов и приложений, так и для работы в качестве коммуникационной шины в микросервисной архитектуре (используется в качестве прокси-сервера для упрощения настройки доступа к сервисам).
3. Redis - резидентная система управления базами данных класса NoSQL с открытым исходным кодом, работающая со структурами данных типа «ключ — значение» (используется для хранения общих данных, например общий ключ для работы SSO).
4. Mongo - документоориентированная система управления базами данных, не требующая описания схемы таблиц; считается одним из классических примеров NoSQL-систем, использует JSON-подобные документы и схему базы данных (используется для хранения объектов сервиса сущностей).
5. Postgres - свободная объектно-реляционная система управления базами данных (используется для хранения реляционных данных сервисов).