Андрей Марков,
директор по развитию технологий компании «ПрограмБанк»

МФО - современные технологии интеграции

ОДНО ИЗ ТОНКИХ МЕСТ ЛЮБОГО СЕРЬЕЗНОГО ПРОЕКТА – ИНТЕГРАЦИЯ. ОСНОВНЫЕ ПРОБЛЕМЫ ВЗАИМОДЕЙСТВИЯ РАЗЛИЧНЫХ РЕШЕНИЙ И КАК ВЫГЛЯДИТ «ПРАВИЛЬНАЯ» ИНТЕГРАЦИЯ МЫ ОБСУДИЛИ С ДИРЕКТОРОМ ПО РАЗВИТИЮ ТЕХНОЛОГИЙ КОМПАНИИ «ПРОГРАМБАНК» АНДРЕЕМ МАРКОВЫМ.

Андрей, насколько сейчас актуальны проблемы интеграции для МФО?

Вопросы «А можно ли выдавать кредиты в вашей программе, а вести бухгалтерию в нашей», или наоборот, нам задавали всегда. Но сейчас на рынке сложилась интересная ситуация:

- С одной стороны, тема интеграции все более актуальна стратегически. Ключевое конкурентное преимущество МФО по сравнению с банком – скорость предоставления займа.
- С другой — эта тема актуальна тактически. В связи с переходом на Единый план счетов с 01.01.2018 идет массовая замена бухгалтерских программ. При этом фронтальный и промежуточный софт, где идет рассмотрение заявок, выдача, обслуживание и взыскание займов, большинство МФО менять не спешат.
- С третьей — все поставщики сейчас декларируют открытость своих систем. Компании объясняют, что их программы можно совместно использовать с любыми другими. Некоторые добавляют, что только, если «та, другая», позволяет. Но дьявол, как всегда, кроется в деталях.

Поэтому сейчас самое время поговорить об этих «деталях» и о том, как вопрос интеграции решен в нашем решении «ПрограмБанк.КредитМикро».

Давайте.

Во-первых, определимся с терминами. «ПрограмБанк.КредитМикро», как и большинство комплексных решений для МФО состоит из двух основных частей: фронт-офиса и бэк-офиса. Фронт-офис автоматизирует работу тех сотрудников, которые непосредственно работают с клиентами. Во фронт-офисе рассматриваются заявки и выдаются займы. Если в МФО ведется сбор просроченной задолженности (и он автоматизирован), то это тоже реализовано во фронт-офисе.

Для принятия быстрых и взвешенных решений о выдаче займов фронт-офис интегрируется с бюро кредитных историй, скоринговыми системами и иными информационными системами, где есть важная информация о потенциальном заемщике, например, с решениями компаний «Кронос-информ», базой МВД украденных паспортов и др.

Для принятия решений о предоставлении кредита юридическим лицам и ИП фронт-офис получает информацию из Единого федерального реестра юридически значимых сведений https://www.fedresurs.ru/#/, информационного ресурса «СПАРК», системы «Интегрум» и др.

Фронт-офис обычно взаимодействует еще с сервисами обмена сообщениями (электронной почтой, SMS-провайдерами, мессенджерами) для информирования клиентов о результатах рассмотрения заявок, дополнительных предложениях, напоминаниях о сроках погашения займа и др.

Бэк-офис — это информационная система, где автоматизирован учет уже выданных займов, расчеты по совершенным операциям, работа бухгалтерии, выпуск отчетности. Бэк-офис связан с фронт-офисом, а также с различными платежными системами, например «Яндек-Деньги», «Киви» и др.

Теперь давайте обсудим саму интеграцию.

Ключевым термином для интеграции современных решений является API – программный интерфейс для приложений. Каждый API – это набор возможностей, которые один программный продукт предоставляет другим. Наглядно это можно представить как некий разъем, розетку, через которую один компонент системы может подключаться к другим. Если такая розетка существует, и мы знаем ее форму и сигналы на различных контактах, то мы можем найти или сделать ответный разъем, подключиться к ней и получать информацию и прочие нужные нам сервисы от другой системы. Могут ли программы взаимодействовать без такой «розетки»? В принципе да, существует альтернативный вариант – пакетный обмен файлами. Относительно недавно он очень широко применялся.

Но обмен файлами имеет два ключевых ограничения:

1. не обеспечивается связь в режиме реального времени;
2. зачастую, требуется участие человека.

Такой подход, например, не может обеспечить принятие решений о выдаче займа в реальном времени или с минимальными задержками. Кроме этого, если МФО имеет ряд офисов, то возникают дополнительные сложности со сбором и распределением пакетных данных. Поэтому предлагаю такую архаику не рассматривать

и вернуться к API.

API обычно описываются в терминах «запрос-ответ». Каждая предоставляемая API возможность (функция, сервис) определяет:

- формат и структуру запросов (входных сообщений), которые внешние приложения могут отправить системе, чтобы воспользоваться API;
- возможные варианты ответов (исходящих сообщений) на эти запросы. Ответы содержат необходимую информацию или отчет о выполненных операциях.

Пример API — интерфейс синхронизации информации о клиентах. Этот API позволяет получить:

- Всю информацию по всем клиентам.
- Только ту информацию по клиентам, которая изменилась (с определенного момента, который задается временной меткой или счетчиком изменений).
- Информацию по заданному списку клиентов.

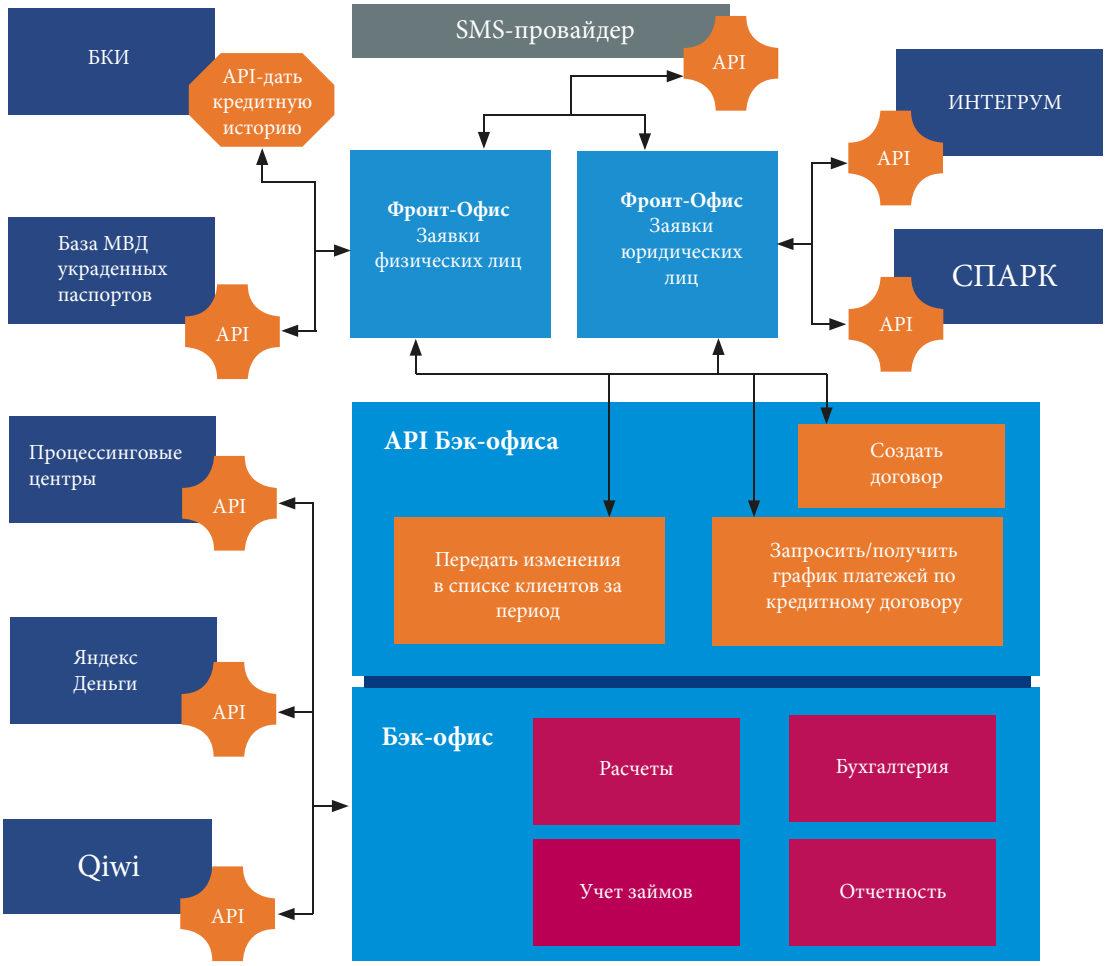
Другой пример API — запрос передачи информации о заключенном договоре в учетную систему. Здесь API не запрашивает существующие данные из системы, а инициирует процесс создания договора в программе учета договоров.

Для обработки запроса «отправить sms на заданный телефонный номер в заданным текстом» sms-провайдер тоже имеет специализированный API, а за информирование о доставке sms, отвечает уже другой API той же информационной системы.

А почему вместо API одна программа не может залезать в базу данных другой программы и непосредственно забирать оттуда информацию (или помещать туда информацию)?

Технически залезть в базу данных другого программного решения не так сложно. Но! Это будет означать прямое вмешательство во внутреннее устройство другого программного продукта. В случае любого изменения в структуре базы данных другого разработчика (а такие изменения происходят постоянно), последствия для работы обеих систем могут быть самыми непредсказуемыми. Также невозможно гарантировать, что при внешнем вмешательстве будут соблюдены все явные и неявные правила, определяющие целостность информации. С точки зрения безопасности данных тоже сложно будет провести четкие границы доступа и ответственности.

Это колоссальный риск, как для разработчика, так и для клиента. Поэтому стабильные документированные API — единственный надежный вариант интеграции. Остальные варианты интеграции могут рассматриваться лишь как временные решения (костыли) в ситуациях форс-мажора.



И какова и ситуация с наличием API в разных программах?

Есть две категории информационных систем, с которыми имеют дело МФО.

Первая категория – информационные системы специализированных поставщиков информационных и финансовых услуг (БКИ, информационные базы, скоринговые системы, системы обмена сообщениями, платежные системы). Они специализируются на предоставлении цифровых сервисов, поэтому их информационные системы имеют все необходимые API. Они подробно документированы и предоставляются на типовых условиях множеству заказчиков.

Безусловно, API дорабатываются и изменяются, но это запланированные и документированные изменения, обычно связанные с расширением функционала или изменениями в законодательстве. При этом четко отслеживаются версии API или отдельных запросов, которые эти API обрабатывают.

Другое дело – взаимодействие разнообразных программ, используемых внутри МФО. Как правило, это описанные выше фронт- и бэк-офисные программы (фронт-офисных программ может быть несколько). Как правило, API предоставляет бэк-офис. Мастер-данные (справочные данные) обычно тоже хранятся именно там – это данные по клиентам, продуктам, тарифам и.т.д. Мастер-данные хранятся в бэк-офисе потому, что туда стекаются данные из всех фронтальных и внешних систем различных производителей и поставщиков. Кроме того, информация в бэк-офисе, как правило, проходит больше проверок и в итоге более корректная.

В такой схеме различные программы могут работать согласованно. Предоставляя API, разработчик гарантирует, что при любых изменениях внутри программ их взаимодействие не нарушится.

Насколько разработка API сложна для разработчика? Является ли предоставление готового API стандартом для нашего рынка?

Вопрос предоставления API делится на две части: техническую и «политическую».

Технически можно реализовать все необходимые API, хотя трудоемкость этого конечно будет зависеть от выбора платформы разработки. «ПрограмБанк. КредитМикро» реализован на современной платформе Java EE, созданной усилиями множества профессионалов специально для разработки критических для бизнеса приложений.

Мы к настоящему моменту реализовали около 40 API, при этом трудозатраты в сумме составили около пяти человеко-месяцев. Они на текущий момент охватывают практически все потребности интеграции с внешними системами, а, в случае необходимости, технологическая база позволит нам оперативно расширить возможности API. При ином выборе платформы разработки это могло бы занять больше времени, но такие инвестиции доступны для любого серьезного разработчика.

«Политическая» часть касается стратегии компании. Если разработчик ориентирован на потребности клиентов и их желание построить для себя наиболее удобный ИТ-ландшафт, то вопросы открытости для него приоритетны. Особенно с учетом текущей ситуации массового обновления бэк-офисных систем. Поэтому у клиентоориентированного разработчика большая часть API уже готова.

Но часть поставщиков бэк-офисных систем не очень заинтересована в предоставлении API. Тем самым они рассчитывают создать условия, чтобы заказчик был вынужден автоматизировать всю деятельность МФО лишь с помощью их программы. С их точки зрения, чем проблематичнее будет выглядеть возможность интеграции с другими системами, тем сложнее заказчику будет решиться на внедрение каких-то программных компонентов от других производителей. К счастью, такой подход встречается на рынке все реже.

В общем, при современном уровне развития технологий, API – это необходимый элемент промышленного решения, и наличие всех необходимых API – хороший критерий как клиентоориентированности поставщика, так и его профессионализма.

А вопрос платформы принципиален для интегрируемости? На каких платформах можно реализовать API?

Современные подходы к созданию API не привязываются к какой-то определенной платформе. Это важное требование, исключающее зависимость от какой-то одной платформы или поставщика. Вопрос лишь в том, насколько платформа соответствует современным стандартам и какие базовые технологии и инструменты в ней изначально реализованы и доступны для использования в приложениях.

Технологии Java EE сейчас на практике являются стандартом для разработки современных корпоративных приложений. В полной мере охватываются этими стандартами и все, широко используемые на практике, средства и протоколы интеграции приложений. Если какие-то более специфические протоколы отсутствуют в стандарте, то реализацию их легко найти в огромном открытом сообществе Java-разработчиков. Поэтому прикладная разработка ведется на базе очень большого разнообразия готовых системных компонентов. В этих компонентах уже решены самые сложные проблемы надежности, производительности и безопасности. Зачастую каркас прикладного кода, в том числе и точки доступа API, генерируется автоматически с помощью этих инструментов. Разработчикам остается реализовать лишь специфические для конкретного приложения элементы бизнес-логики и связать точки доступа API с внутренними функциями и данными приложения.

Если в выбранной платформе такие базовые системные компоненты присутствуют не в полной мере, то реализовать их на должном уровне в рамках какого-то отдельного проекта или решения весьма проблематично.

Допустим, что все же поставщик бухгалтерского решения медленно реализует необходимые API. Можно ли организационно упростить требуемый объем интеграции?

Частично, например, если печатать договор в бэк-офисе (а не во фронт-офисе), то количество необходимых API уменьшится: не надо будет, к примеру, импортировать график погашения из бэк-офиса, где он рассчитывается. Но это усложнит работу кредитных менеджеров, поскольку им придется переключаться между разными программными продуктами.

Правильно ли я понимаю, что проблема интеграции, это исключительно проблема наличия API?

Нет, наличие API для интеграции – это лишь необходимое условие, но не достаточное.

Во-первых, весь обмен информацией в штатном режиме должен выполняться автоматически, без участия человека либо в режиме реального времени, либо по расписанию.

Участие человека необходимо только в том случае, если в процессе обмена информацией произошел неустранимый сбой. В этом случае система должна автоматически диагностировать ситуацию и информировать о ней администратора.

Во-вторых, необходима не только согласованная работа различных информационных систем в рамках выполнения отдельных функций, но и связывающие эти функции воедино сквозные бизнес-процессы, приводящие к достижению определенных бизнес-целей и показателей. Например, необходимо автоматически принимать решение о предоставлении кредита на основе информации, полученной из нескольких БКИ и других информационных систем. Для решения этой задачи в «ПрограмБанк.КредитМикро» может быть реализована, например, следующая схема:

«ПрограмБанк.КредитМикро» запрашивает первое наиболее доступное или информационно емкое БКИ, и, если у заемщика плохая кредитная история, то последующие БКИ уже не запрашиваем, не тратим лишних денег. Если у заемщика все в порядке, то направляем запрос в следующее БКИ, при получении положительного результата запрашиваем базу МВД по украденным паспортам и др. Если не обнаруживается негативная информация, то принимаем решение о предоставлении займа, и информируем заемщика об этом посредством SMS.

Таким образом, получаем одобрение заявки в полностью автоматизированном режиме. Такая технология часто используется для краткосрочных займов на небольшие суммы. Например, для займов до зарплаты.

Правильно ли я понимаю, что для полностью автоматизированного режима все взаимодействия между разными решениями должны происходить в online режиме?

Использование сторонних сервисов (БКИ и др) на этапе рассмотрения заявки, конечно, должно проходит в online-режиме. Заключенный договор передается в бэк-офис тоже незамедлительно. Сразу могут быть проведены платежи по выдаче займа. А вот информацию о внесении изменений в данные о клиентах можно передавать из бэк-офиса во фронтальную часть периодически, даже раз в сутки.

В каких-то случаях частота взаимодействия определяется регламентом внешних систем. Например, платежная система «Киви» присылает данные реестра платежей раз в сутки, причем, просто в виде файла по электронной почте. «КредитМикро» самостоятельно диагностирует, что поступила новая информация, и обрабатывает ее – тоже в автоматическом режиме.

Разве бизнес-логика при обмене информации реализуется не в интеграционной шине?

Не обязательно. Если используется интеграционная шина, она может взять на себя управление информационными потоками между системами. В большинстве МФО, с которыми мы работаем, интеграционные шины не используются. В этом случае логика взаимодействия между информационными системами МФО и внешними информационными сервисами настраивается с помощью Workflow в «ПрограмБанк.КредитМикро».

Подытоживая наш разговор:

- Современные технологии интеграции позволяют организовать совместную работу различных программных продуктов как единого целого, для определенных бизнес-процессов, без участия человека.
- Логика взаимодействия различных систем должна определяться либо в одной из интегрируемых систем, выбранной в качестве базовой и обладающей соответствующими возможностями, либо в специальной интеграционной платформе.
- Необходимое условие устойчивой интеграции – наличие стабильных API.