BaaS платформа — революция в архитектуре IT банковской системы

А. П. Романчук— генеральный директор компании Цифровая динамика.

Преамбула (откуда появилась идея Платформы)

В недавно вышедшей резонансной статье[1] мы в общих чертах изложили свою позицию о том, как должен быть устроен современный цифровой банк. Там отражён перечень проблем, с которым мы сталкивались и продолжаем сталкиваться, запуская банковские проекты в течение последних 25 лет. Полагаю, что имеет смысл на них более подробно остановиться, чтобы было понятно, почему сформировалась такая позиция.

В стародавние времена, когда у нас был только один банк («Северная казна»), мы по полной программе вкусили результат стратегии «всё-в-АБС», когда абсолютно все разработки пытались «впихнуть» именно туда – полное досье на клиента и CRM, электронные подписи, скоринг-модели, кредитные процессы, даже элементы документооборота, расчеты всех комиссий и пр. Это была АБС «Банкир» (ИБС Банкир на СУБД Progress) и можно, конечно, много говорить о том, насколько это несовременное решение, но тем не менее, оно максимально близко тем старым «мейнфремовым» АБС, которые мы упомянули в статье. Интеграция с ней была запредельно неудобная (фактически файлообмен), пользовательские интерфейсы в терминале в духе FoxPro 2.0 для MS-DOS. Автоматическое тестирование, репозиторий исходных кодов, CI/CD при таком подходе были совершенно невозможны, а в результате, общей проблемой было то, что постоянно что-то ломалось, выпускались некачественные разработки.

Всё это породило то, что банк фактически стал зависим от нескольких АБС-программистов, которые все трудились в режиме вечного аврала, многие из которых уже тогда были предпенсионного возраста, а нормального человека затащить в дебри языка Progress 4GL (язык разработок в АБС) было можно только от безысходности (эта история очень сильно похожа на упомянутые байки про COBOL-программистов). Пытались даже решать проблему путем прикручивания хранилища/шины (ESB) за кучу денег – но выбросили деньги на ветер, поскольку ничего из ожидаемого не заработало. Горизонтальному масштабированию СУБД от АБС в принципе не подлежало, и акционеры банка были вынуждены тратить значительные средства на покупку всё более мощной центральной «железяки» для поддержания работы АБС.

Решать проблему АБС пытались в том числе за счет того, что начали строить параллельные IT-системы – отдельно документооборот, отдельно скоринг-сервис. В какой-то момент стало понятно, что на стороне ДБО- системы мы должны держать значительно больше всякой бизнес-логики – проверки, остатки, интеграции с карточным процессингом и иными системами, поскольку в АБС это было добавлять сложно, долго и работало нестабильно, прежде всего из-за отсутствия современных инструментальных средств (автотесты, очереди, распределенные транзакции и пр.).

Когда в 2004 году начался процесс выбора новой АБС, то первая задача была в том, чтобы, как минимум, мигрировать на АБС на Oracle, поскольку по Oracle было больше специалистов на рынке (как по администрированию, так и по разработке). Выбор шел между Forbis Forpost (Литва), Temenos, Кворум NEXT. В частности, в Temenos

[1] https://plusworld.ru/journal/2025/plus-6-326-2025/it-modernizatsiya-v-stile-trishkinogo-kaftana-ili-kak-bankam-spravitsya-s-sovremennymi-vyzovami-na-b/

(который вроде бы уже тогда мог работать на Oracle, а не только jBase) я лично увидел в первую очередь машину для генерации проводок и всяких операций по завершению операционного дня без блокировки системы (например, у нас было много кредитов, и наша АБС Банкир еле справлялась с этой обработкой, блокируя всю операционную работу АБС на целую ночь, мешая нам нормально масштабироваться по часовым поясам). Но тогдашнее руководство IT-банка повелось на маркетинг от Кворум NEXT, где внутри АБС якобы можно было рисовать красивые схемы бизнес-процессов. Однако, внедрение тормознулось по причине того, что в новую АБС перетащить всё оказалось сложно, проект затягивался, приостановился, а потом банк «Северная казна» был продан Альфа-банку в 2008 году.

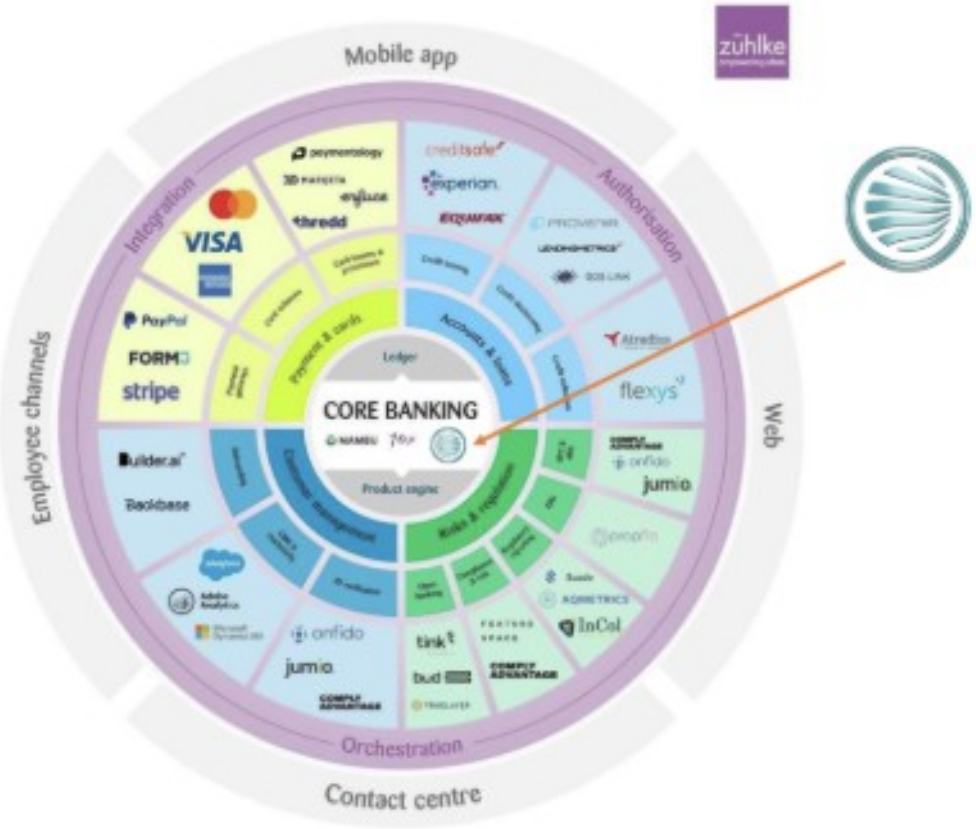
В новом банковском проекте («Инбанк») мы более осознанно подошли к архитектуре и потратили значительное время, чтобы выбрать себе АБС на старте (банк был «нулевой», клиентов почти не было). Основная идея, которую хотели положить в основу – взять АБС в как можно более стандартной комплектации, делать минимальные настройки, которые бы относились только к бухучету, а всю логику работы с продуктами строить за пределами АБС, на серверах приложений ДБО. Примерно в 2011 году у нас созрела идея Banking-as-a-Service (BaaS), когда мы начали решать задачи интеграции нашего банка через API с внешними клиентами (т. е. как Software-as-a-Service, но только применительно к банку). АБС у нас была ЦФТ в стандартной комплектации, и мы гордились тем, что у нас в штате было всего полтора разработчика АБС, а любые попытки что-либо затянуть на сторону АБС жестко пресекались. Однако, именно на примере ЦФТ мы смогли убедиться, что такое «АБС-мафия», когда разработчики проприетарной АБС формируют вокруг себя сообщество сертифицированных разработчиков, создают рынок таких специалистов, которые в дальнейшем выкручивают руки менеджменту банка, заставляя платить им огромные деньги и непрерывно шантажируя тем, что собираются уйти. Совсем недавно мы общались с несколькими топ-менеджерами банков в РФ, которые пострадали от рук «АБС-мафии».

Платформа «Инбанка», разработанная нами, была продана в 2013 году в Банк24.Ру и легла в основу позднее запущенного в январе 2015 года банка «Точка». Принципы разделения между ДБО и АБС помогли нам очень быстро и легко переключиться на новую АБС, которая была в банке, а мощное ядро с элементами API, заложенные средства масштабирования и отказоустойчивости позволили Точке очень быстро запуститься, сосредоточиться в основном на пользовательском опыте.

После 2014 года начали работать на зарубежных рынках: проект электронного кошелька на блокчейн Copernicus Gold, совместные проекты со стартапами JPMorgan Chase в области возобновляемой энергетики, работа на рынке драгметаллов в Сингапуре, пилотный проект по p2p-кредитованию в Уганде и краудфандинг-платформа в Европе.

В итоге, когда в 2022 году выпускали новую платформу DW-DYNAMICS, мы уже четко сформировали описанную в статье модель Финансовая организация/Банк = Учет/Отчетность + BaaS + UI, четко разделяя бизнес-функции продуктов от их учета.

Есть определенная терминологическая путаница, на наш взгляд, из-за того, что считать Core Banking, а что не считать. Нам нравится вот такая картинка, которая показывает, что на уровне Core Banking есть определенное разделение сочетаний Ledger и Product Engine.



Мы как платформа позиционируем себя скорее как Product Engine, нежели как Ledger, т. е. главная книга, хотя для многих случаев мы также предоставляем и то, и другое.

Также отдельно где-то рядом с Ledger выделяем слой формирования отчетности перед регуляторами, но стараемся в эту тему не лезть, поскольку она, как правило, специфичная для каждой страны, отрасли и часто там очень много постоянных изменений, так что проще использовать какое-то имеющееся на рынке отдельное заточенное решение, где кто-то сам следит за необходимыми обновлениями.

В итоге наше позиционирование выглядит так – мощное ядро для современного цифрового банка, его продуктовый движок (Product Engine), выставляющий наружу высокоуровневое API (Banking-as-a-Service) и в некоторых случаях берущее на себя функции в том числе главной книги (Ledger).

В такой схеме кто наши клиенты:

1. Банки, которые хотят построить мощный цифровой банк с API и работать в сегментах embedded finance/banking, Open API, Integration Banking, Fintech-as-a-Service (агрегатор финтехов) (**строят цифровой банк с API**);
2. Банки, которые хотят значительно уменьшить зависимости от своего АБС-вендора по причине того, что надоело постоянно платить ему за любые доработки, и свести его участие лишь к стандартным операциям, а всю продуктовую разработку вести отдельно современными средствами - писать микросервисы, сосредоточиться на UI/UX (**хотят оптимизировать свой внутренний движок**);
3. Различные финтехи (практически любой сферы – банковские, токенизация, брокеры, электронные кошельки и пр.), причем в таких проектах мы готовы брать на себя всю работу по содержанию и доработкам под конкретный

проект, чтобы наш клиент мог полностью сосредоточиться на бизнесе, не заморачиваясь об IT-составляющей (**хотят начать новый финтех-бизнес**).

Основные принципы Платформы

При проектировании своей платформы мы исходили из следующих принципов, которые должны быть в неё заложены:

1. **Универсальность** – мы должны уметь запускать на платформе вообще любой финансовый продукт, в любой юрисдикции и рассчитанный на любую аудиторию (c2c, b2c, b2b).
2. **API-first** – любая операция в системе должна быть реализована как API-функция, т.е. API должно быть не что-то, приделанное сбоку, а некая базовая сущность платформы.
3. **Платформа-конструктор** – мы закладываем в платформу различные проработанные кубики, из которых легко и быстро собираем любой проект.
4. **Легкая кастомизируемость** под любой проект – мы должны уметь добавлять/менять функционал в любом месте так, чтобы получалась нужная нам функциональность.
5. **Мультипроектность** – на одной инсталляции платформы может работать много абсолютно разных проектов (White Label) – разные банки, разные финтехи, которые имеют собственные клиентские базы, транзакции, интеграции и не пересекаются.
6. **Продвинутые схемы безопасности** – ролевые, матричные модели доступа, доступ в зависимости от полномочий, разных каналов, электронные подписи, защита от администраторов.
7. **Токенизация** - единообразная работа с любой ценностью: валюты, драгметаллы, удобрения, электроэнергия и так далее. За счет токенизации работаем с чем угодно.
8. Платформа должна реализовывать в себе **функции** и **шины данных** за счет встроенного движка очередей и иметь механизм различных коннекторов, а также позволять строить простые бизнес-процессы и обеспечивать жизненный цикл финансовых документов.
9. **Поддержка собственной главной книги** с проводками – система должна иметь встроенный журнал проводок во внутреннем собственном плане счетов, но с возможностью конвертации из него в любую другую систему учета. Клиентские выпискидолжны строить по своему журналу проводок, обеспечивая максимальную производительность. Возможны варианты, когда проводки выгружаются во внешнюю системы полностью (например, при использовании в качестве внешней главной книги АБС банка) или же в агрегированном виде (например, в варианте готовом для загрузки в бухгалтерские системы типа Xero).

Функции и модули

Модульность платформы построена по функциональному принципу, а не продуктовому. Имеются следующие модули:

1. Ядро системы. Содержит все базовые объекты и функции, в том числе подсистему безопасности.
2. Сервер авторизации. Управление пользователями и их аутентификаций, поддержка протокола OAuth 2.

3. API-сервисы – доступные через REST протокол функции системы и публикуемые в спецификации[2].
4. Главная книга.
5. Подсистема для маршрутизации сообщений через очереди (встроенная шина)
6. Модуль асинхронных задач и задач по расписанию.
7. Коннекторы для банка и различных внешних финансовых и нефинансовых сервисов.
8. Сервисы мгновенных нотификаций по протоколу WebSocket.
9. Web-приложение BackOffice для работы сотрудников финансовой компании (функции CRM, клиентские операции и процессы, администрирование системы, просмотр отчетов и пр. информации).

Готовые кастомизации

В настоящий момент в Платформе есть несколько готовых кастомизаций под различные виды проектов, например:

1. Расчетный мультивалютный банк - все виды платежей, юридические и физические лица, любые продукты: депозиты, кредиты, инвестиционные продукты;
2. Оператор мультивалютных электронных кошельков - открытие мультивалютных кошельков, платежи, ввод/вывод средств, интеграция с другими платежными системами.
3. Оператор цифровых финансовых активов (токенов) – эмиссия токенов под какое-либо обеспечение, транзакции с участием токенов, сделки покупки/продажи, ввод/вывод средств.
4. Брокер драгметаллов: полный цикл работы с металлом – прием и оценка лома, покупка/продажа, конвертация в физический вид, доверительное хранение металла, трейдинг под залог металла.
5. Токенизации возобновляемой электроэнергетики - эмиссия токенов, обеспеченных кВт-ч, выпуск сертификатов, доказательство генерации, сделки покупки/продажи электроэнергии.
6. Оператор краудфандинговой площадки.
7. Сервис взаиморасчет – выявление цепочек дебиторской задолженности и расчеты по ним.
8. Трансграничные платежи через любой стабильный актив (корзину активов) – например, золото.
9. Система расчетов через бартерные цепочки.
10. Банк-оператор маркетплейсов.

Заключение.

При использовании BaaS-платформы финансовая организация, банк сразу фокусируется на своих продуктах и бизнесе, а не теряет деньги и время на многолетние внутренние разработки и тестирования IT-решений. Самые продвинутые финансовые организации, особенно в США, значительно сокращают прежде огромный штат своих IT-программистов, концентрируясь наспециалистах-скаутов технологий, которые выбирают на рынке самые передовые IT-решения и интегрируют их в свою архитектуру. Всовременном мире выигрывает не тот, кто больше всего вкладывает финансовых и людских ресурсов в развитие, а тот, кто быстрее внедряет новейшие технологии.

---

[2] https://dw-dynamics.stoplight.io/docs/dwd-api-bank

Требуется ли уникальная квалификации персоналу банков для перехода на технологию BaaS? Безусловно, нет. Современные платформы очень просты для пользователя, для IT-специалистов и банковского персонала. Проще, чем их нынешнее программное обеспечение. Было бы желание у акционеров и руководителей банков двигаться в ногу с прогрессом в финансовой сфере и не терять, а укреплять свои рыночные позиции.