ПЕРША CRM СИСТЕМА

Система управління процесами, розподілене сховище та телекомунікаційний фреймворк для побудови систем документообігу

Максим Сохацький

Київ, 12 жовтня 2021

Зміст

Розділ 1

Вступ

Система автоматизації підприємства ERP.UNO визначає формальну бізнес специфікацію та її імплементацію для сучасних оптимізованих підприємств які прагнуть стабільності та довершеності, не втрачаючи при цьому повноти визначеності та відповідності реальним умовам та потребам. Телекомунікаційна платформа Erlang/OTP від Ericsson успішно застосовується в індустрії мобільними операторами понад 30 років, а її віртуальна машина досі вважається однією з найкращих в галузі. Системи ERP на її базі також уже не один рік використовується у банківській сфері, процесінгу транзакцій, розподілених системах повідомлень, в IoT секторі. Ви можете переглянути демо модулі системи ERP.UNO в нашому захищеному середовищі n2o.cloud зі своїм центром випуску ECC X.509 сертифікатів. На цій сторінці ви знайдете перелік модулів системи та основні сутності схеми.

1.1 Модулі підприємства

SYNRC Підприємство є комплексом бібліотек160;(N2O.DEV) та підсистем додатків160;(N2O.SPACE), який використовує загальну шину і загальну розподілену базу даних.Перша глава дає огляд існуючих рішень та вступ до предмету формальної верифікації.

LDAP — Сервер аутентифікації, зберігання ключів та директорія підприємства.

ERP — Даний модуль зберігає основну ієрархічну структуру підприємства, її схему, записи про персонал, інвентар, компанії та офіси підприємства.

FIN — Фінансовий модуль підприємства, зберігає бізнес процеси, які представляють собою рахунки учасників системи: персонал (для нарахування зарплат), рахунки та субрахунки підприємства (для здійснення економічної діяльності) і зовнішні рахунки в платіжних системах.

ACC — Система управління персоналом: зарплатні відомості, календар підприємства, відпустки, декретні відпустки, інші кален-

дарі.

SCM — Система управління ланцюжком поставок: головний БП системи — експедиційний процес доставки товарів ланцюжку одержувачів за допомогою транспортних компаній.

CRM — Система управління клієнтами: являє собою розширення більш абстрактного додатку CHAT.

PLM — Система управління життєвим циклом проектів і продуктів. Також містить CashFlow та P38;L звіти.

PM — Система управління проектами підприємства з деталізацією часу і протоколів прийому-передачі (прийняті коміти в гитхабі).

WMS — Система управління складом.

TMS — Система управління транспортом підприємства.

1.2 Управління ресурсами

Головним чином інформаційна структура нашого підприємства складається з обчислювальних ресурсов (додатки, запущені в шині) та накопичувальних ресурсів (дані, збережені в базі даних).

SOA архітектура в якості моделі управління обчислювальними ресурсами пропонує асинхронний протокол віддаленого виклику на шинах. Разом з N2O можно використовувати MQTT та інші шини, за допомогою наступних протоколів: TCP, WebSocket. Ці асинхронні протоколи часто називають протоколами реального часу, оскільки в них функції відправки повідомлень завжди миттєво повертають результат. Що ж стосується протоколів для публікації і доступу до даних, то тут може виявитися доречним використання синхронного HTTP протоколу.

1.3 Обчислювальні ресурси

Для SOA архітектури традиційно використовуються асинхронні протоколи доступу до обчислювальних ресурсів. Зазвичай це серверні воркери, які підключені до шини і обслуговують API певного додатку. Кожен додаток має власне консистентне хеш-кільце воркерів. В мережі одночасно працює багато кілець-додатків.

```
config :n2o,
  tcp_services: ['ldap'],
  ws_services: ['chat'],
  mqtt_services: ['erp', 'bpe']
```

За допомогою config.exs файлу можна налаштувати необхідну конфігурацію серії консистентних кілець, кожне з яких працює на власному транспортному протоколі. В даному прикла-

ді показано карту Erlang серверів, які обслуговують черги додатків в шині:

```
> PLM.vnodes
[
  {{:tcp, '/ldap/tcp/4'}, [:n2o_tcp]},
  {{:tcp, '/ldap/tcp/3'}, [:n2o_tcp]},
  {{:tcp, '/ldap/tcp/2'}, [:n2o_tcp]},
  {{:tcp, '/ldap/tcp/1'}, [:n2o_tcp]},
  {{:ws, '/chat/ws/4'}, [:n2o_ws]},
  {{:ws, '/chat/ws/3'}, [:n2o_ws]},
  {{:ws, '/chat/ws/2'}, [:n2o_ws]},
  {{:ws, '/chat/ws/1'}, [:n2o_ws]},
  {{:mqtt, '/erp/mqtt/4'}, [:n2o_mqtt]},
  {{:mqtt, '/erp/mqtt/3'}, [:n2o_mqtt]},
  {{:mqtt, '/erp/mqtt/2'}, [:n2o_mqtt]},
  {{:mqtt, '/erp/mqtt/1'}, [:n2o_mqtt]},
  {{:mqtt, '/bpe/mqtt/4'}, [:n2o_mqtt]},
  {{:mqtt, '/bpe/mqtt/3'}, [:n2o_mqtt]},
  {{:mqtt, '/bpe/mqtt/2'}, [:n2o_mqtt]},
  {{:mqtt, '/bpe/mqtt/1'}, [:n2o_mqtt]},
  {{:caching, 'timer'}, [:n2o]}
]
```

Завдяки такій деталізації можна проектувати гетерогенні системи, включаючи необхідний набір протоколів на портах потрібних машин. Ця же система дозволяє отримати балансування навантаження, підключаючи фізичні ресурси до певних черг шини даних.

В нашій моделі асинхронні протоколи використовуються для управління обчислювальними ресурсами підприємства.

1.4 Накопичувальні ресурси

Розподілені хеш-кільця використовуються не тільки для розподілених обчислень, але і для зберігання даних. Деякі бази даних, наприклад RocksDB та Cassandra, використовують глобальний простір ключів для даних (на відміну від таблично-орієнтованих баз). Саме для таких баз і створено библиотеку KVS, де в якості синхронного транзакційного інтерфейсу — API ланцюжків з гарантією консистентності. Нижче наведено приклад структури ланцюжків екземпляру системи PLM:

```
> :kvs.all :writer
[
  {:writer, '/bpe/proc', 2},
  {:writer, '/erp/group', 1},
  {:writer, '/erp/partners', 7},
  {:writer, '/acc/synrc/Kyiv', 3},
  {:writer, '/chat/5HT', 1},
  {:writer, '/bpe/hist/1562187187807717000', 8},
  {:writer, '/bpe/hist/1562192587632329000', 1}
]
```

В нашій моделі синхронні протоколи використовуються для управління накопичувальними ресурсами підприємства і транзакційного процесингу.

1.5 Типові специфікації

Протоколи визначаються типовими специфікаціями і генеруються для наступних мов: Java, Swift, JavaScript, Google Protobuf V3, ASN.1. Також ми генеруємо валідатори даних по цих типових анотаціях і вбудовуємо ці валідатори в тракт наших розподілених протоколів, тому ми ніколи не дозволимо клієнтам зіпсувати сторадж. Для веб додатків у нас развинута система валідації — як для JavaScript, так і на стороні сервера. Бізнес логіка повністю ізольована в нашій системі управління бізнес процесами, де кожен бізнес процесс є процесом віртуальної машини. Всі ланцюжки модифікуються атомарним чином, підтримують flake адресацію, і не вимагають додаткової ізоляції у своєму примітивному використанні. Тому ви можете трактувати базу як розподілений кеш і використовувати її з фронт додатків для примітивних випадків.

Розділ 2

Загальна структура

У цій статті я розповім, з чого складається модуль підприємства.

Якщо розглядати первинну гранулярність підприємства, яке в першому наближенні складається з модулів, то уже на прикладі PLM ми можемо виявити повну структуру типового модуля, кожен елемент якої представлений в екземплярі PLM.

2.1 Конфігурація

Перший і головний компонент додатка — файл конфігурації (для Erlang — sys.config, для Elixir — consig.exs), який потрібен для багатьох додатків-залежностей: n2o, kvs, erp, form. Це обов'язковий компонент будь-якого ерланг додатка, який потребує ці залежності.

Більш детально про конфігурацію Erlang і Elixir додатків можна почитати тут:

— <a href="https://erlang.org»Erlang
 — <a href="https://elixir-lang.org»Elixir

2.2 Публикація

Для побудова релізу, звичайного запуску або публікації в hex.pm за допомогою mad, mix чи rebar3, вам необхідний файл публікації (для Erlang — rebar.config, для Elixir — mix.exs). Файл публікації містить план запуску додатків.

Більш докладно про публікацію Erlang і Elixir додатків можна почитати тут:

— <a href="https://mad.n2o.space»mad (Erlang)
 — <a href="https://www.rebar3.org»rebar3 (Erlang)
 — <a href="https://elixir-lang.org/getting-started/mix-otp/introduction-to-mix.html»mix (Elixir)

2.3 Типові специфікації

Типова специфікація — це сукупність визначень типів (type), записів (record) та специфікацій функцій (spec). Це информація для діалайзера, який допомагає визначити невідповідність коду цим специфікаціям. Всі системи збірки підтримують перевірку dialyzer.

У типових специфікаціях ми зберігаємо внутрішні структури фреймворків і додатків, а також бізнес-об'єктів підприємства. Мова опису бізнес об'єктів підтримує кортежі (для повідомлень), суми (для протоколів), скалярні і векторні типи (для полів).

Типові специфікації зберігаються в HRL файлах, в папці include. Тут повинні бути всі -spec, -record, -type визначення. В Elixir імпортуйте їх за допомогою Record.extract.

Якщо програма не містить include папки (наприклад, як PLM модуль), то це означає, що модуль не визначає ніяких додаткових типів, а користується типами своїх залежностей, або не користується ними взагалі.

Більш докладно про типові специфікації та підтримувані мови програмування можна почитати тут:

— <a href="https://bert.n2o.space»bert

2.4 Протоколи

Якщо додаток реалізує якийсь протокол, цей протокол вбудовується в протокольні цикли $\mathrm{n2o_m qtt, n2o_w s, n2o_t cp, .}$

Список протоколів визначається у змінній protocols бібліотеки N2O:

```
protocols:
[
  :n2o_heart,
  :n2o_nitro,
  :n2o_ftp,
  :bpe_n2o,
  CHAT.TXT
]
```

А список воркерів, які реалізують ці протоколи — на ендпоінтах:

```
mqtt_services: ['erp', 'plm'],
ws_services: ['chat'],
```

Протоколи, якщо вони реалізовані додатком, знаходяться в папках src/protos і lib/protos для Erlang і Elixir відповідно.

Більш докладно про N2O протоколи та їх використання можно почитати тут:

— <a href="https://ws.n2o.space»n2o

2.5 Ланцюжки

Все типізовані типовими специфікаціями дані зберігаються в KVS сховищі. Це Erlang-орієнтована абстракція над записами/кортежами (records, tuples, C-structures), яка дозволяє приховувати за єдиним інтерфейсом декілька KV сховищ (включаючи Mnesia, RocksDB, Cassandra).

2.6 Кортежі і їх ланцюжки

В основі KVS лежать поняття кортежа і ланцюжка. Типи кортежів визначаються в типових специфікаціях, а ланцюжки — це послідовності кортежів, в загальному випадку будь-яких типів, таким чином можна говорити про гетерогенні та гомогенні ланцюжки.

Кожен ланцюжок індексується своїм ідентифікатором, який представляє собою сегментований шлях в ієрархічній віртуальній файловій системі. Це зроблено для того, щоб префіксним пошуком можна було вибрати всіх дітей певної субгілки в ієрархії ідентифікаторів ланцюжків. Всі ідентифікатори всіх ланцюжків також знаходяться в ланцюжку.

2.7 Схеми

Кожен модуль підприємства може включати одну чи багато схем. Схема — це сукупність типових специфікацій, іншими словами — певний набір кортежів і їх типів, як форма дистрибуції типової специфікації.

2.8 Первинні кореневі ланцюжки

Щоб не створювати руками всі базові словники і основні організаційні структури, зручно винести їх в так звані завантажувальні модулі. Ці записи автоматично створюються при холодному старті додатку ERP. Кореневим первинним ланцюжкам присвячені одразу дві наступні частини: Частина 3. Створення первинних ланцюжків, де розповідається, як створювати організаційну структуру підприємства у вигляді завантажувальних модулів первинних кореневих ланцюжків. і Частина 4. Створення адміністратора даних, де розповідається, як створити універсальний переглядач ланцюжків у вигляді окремого модуля підприємства.

Більш докладно про систему зберігання KVS та управління типовими специфікаціями можна почитати тут:

— <a href="https://kvs.n2o.space»kvs

2.9 Процеси

Якщо всі дані інформаційної системи підприємства зберігаються в ланцюжках, то еволюція цих даних відбувається за допомогою бізнес-процесів. Бізнес-процеси покликані вирішити певні проблеми, пов'язані з масштабуванням бізнес-логіки на виробництві, тому ця частина підприємства добре стандартизована з 2008 року, з появою більш-менш універсального стандарту BPMN, який частково підтримується системою управління процесами BPE.

У загальному випадку бізнес процеси (БП) - це графові уявлення алгоритму з іменами переходів, станів і асоційованих функцій. Всі бізнес-модулі підприємства реалізують якийсь головний БП, і серію допоміжних процесів. Покликанням БП є вирішення проблеми ізоляції розподіленої транзакції у вигляді окремого процесу віртуальної машини. Цей БП являє собою звичайну функцію action/2, аргументами якої є ідентифікатори ланцюжків. У якості ефектів цей БП генерує дані в інших ланцюжках, реалізуючи таким чином обчислювальну модель обчислення процесів.

Наприклад, БП "Рахунок в Банку"є циклічним рекурентним процесом, який виходить з, і входить в один і той же стан (моноїд). В якості аргумента у цієї функції, що складається з однієї умови, є тільки скалярна величина — бізнес об'єкт "Транзакція". Таким чином, трейс цього процесу — ланцюжок транзакцій. Операція переказу грошей в такій моделі означатиме розподілену транзакцію між усіма учасниками переказу, контрольовану окремим процесом.

У системі, яку розглядаємо, модуль PLM включає три процеси: 1) Процес "Рахунок в Банку"фінансового модуля FIN; 2) Процес "Продукт"модуля PLM; 3) Процес "Пре-Продукт"модуля PLM. В Частині 5. Адміністратор процесів показано, як створити адміністратора процесів для модуля BPE, призначеного для ознайомлення з системою, а також для примітивного ручного тестування.

Більш докладно про систему управління бізнес-процесами BPE та її використання можна почитати тут:

— <a href="https://bpe.n2o.space»bpe

2.10 Сторінки

2.11 Редактори

<figure></figure>

2.12 Вектори

2.13 Роутери

Більш докладно про веб-фреймворк NITRO можна почитати тут:

2.14

Розділ 3

Створення ланцюжків

У цій статті ми покажемо, як визначати схеми бізнес-об'єктів підприємства та ієрархічну структуру ідентифікаторів ланцюжків цих об'єктів. Це визначає консольну взаємодію користувача та інформаційної системи.

</section> <section>

3.1 Типові специфікації

Схема включає у себе всю типову інформацію про бізнес-об'єкти. Схема ERP модуля визначає базові об'єкти організаційної структури, об'єкти платіжної системи PAY та систему обліку співробітників ACC.

<h4>ERP SPEC — Таблиці организаційної структури</h4>

```
-type locationType() :: normal | extra.

-record('Loc',
{ id          = kvs:seq([],[]) :: [] | term(),
  code        = [] :: [] | term(),
  country     = [] :: [] | binary(),
  city        = [] :: [] | binary(),
  address     = [] :: [] | binary(),
  type        = [] :: locationType() }).

-record('Branch',
{ id          = kvs:seq([],[]) :: [] | term(),
  loc         = [] :: [] | #'Loc'{} }).

-record('Inventory',
{ id          = [] :: [] | binary(),
  name        = [] :: [] | binary(),
  branch      = [] :: [] | #'Branch'{},
  type        = [] :: term() }).

-record('Organization',
{ name        = [] :: [] | binary(),
  url         = [] :: [] | string(),
  location    = [] :: [] | #'Loc'{},
  type        = [] :: term() }).
```

```
-record('Person',
{ id          = kvs:seq([],[]) :: [] | term(),
  cn          = [] :: [] | binary(),
  name        = [] :: [] | binary(),
  displayName = [] :: [] | binary(),
  location    = [] :: #'Loc'{},
  type        = [] :: term() }).


-record('Employee',
{ id          = kvs:seq([],[]) :: [] | binary(),
  person      = [] :: [] | #'Person'{},
  org         = [] :: [] | #'Organization'{},
  branch      = [] :: [] | #'Branch'{},
  type        = [] :: term() }).
```

<h4>PAY SPEC — Таблиці системи управління платежами</h4>

Гроші зберігаються у форматі N,M, де N — кількість знаків після коми, а M — усі значущі цифри. Таким чином числа кодуються багатьма способами, наприклад одиниця: 1 = 0,1 = 1,10 = 2,100. Операція множення у такій системі виглядає просто mul(A,B,C,D) -> A+C,B*D.

```
-type fraction_length() :: integer().
-type digits() :: integer().
-type money() :: {fraction_length(),digits()}.

-record('Payment',
{ invoice = [] :: [] | term(),
  volume = [] :: [] | money(),
  price = {0,1} :: money(),
  instrument = [] :: term(),
  type = [] :: paymentType(),
  from = [] :: term(),
  to = [] :: term() }).
```

<h4>PLM SPEC — Таблиці системи управління життєвим циклом</h4>

```
-record('Acc',
{ id   = [] :: [] | binary() | list(),
  rate = {0,0} :: money() }).

-record('Product',
{ code          = [] :: [] | term(),
  id            = kvs:seq([],[]) :: [] | binary(),
  url           = [] :: [] | binary() | list(),
  engineer      = [] :: [] | #'Person'{},
  director      = [] :: [] | #'Person'{},
  owner         = [] :: [] | #'Person'{},
  organization  = [] :: [] | #'Organization'{},
  type          = [] :: productType() }).

-record('Investment',
{ id = [] :: [] | term(),
  volume = [] :: [] | money(),
```

```
price = {0,1} :: money(),
instrument = [] :: term(),
type = [] :: investmentType(),
from = [] :: term(),
to = [] :: term() }).
```

</section> <section>

3.2 Створення кореневих ланцюжків

ERP BOOT або пусконаладка підприємства — це процес заповнення первинних словників та таблиць фундаментальною інформацією. Головним чином це відображення ієрархічної, організаційної структури підприємства. Від працівника підприємства, його рабочого місця, його локального офісу, його локальної компанії, і далі, до группи міжнародних компанії з офісами в різних країнах світу, та можливо, навіть до синдикатів транснаціональних корпорацій.

<h4>ERP BOOT — Організаційна структура підприємства</h4>

Розглянемо приклад: компанія Quanterall, головний підрядник Aeternity, має офіси в Софії, Варні (головний офіс компанії) та Пловдіві. Сама компанія здійснює операції тільки в Болгарії, тому группа складається из однієї компанії.

Добавляємо, зараз та надалі, дані за допомогою list комбінаторів:

```
-module(erp).
-compile(export_all).

boot() ->
    GroupOrgs =
  [ #'Organization'{
      name="Quanterall",
      url="quanterall.com"} ],

    HeadBranches =
  [ #'Branch'{ loc = #'Loc'{ city =
     "Varna", country = "BG" } },
    #'Branch'{ loc = #'Loc'{ city =
     "Sophia", country = "BG" } },
    #'Branch'{ loc = #'Loc'{ city =
     "Plovdiv", country = "BG" } } ],

    PartnersOrgs =
  [ #'Organization'{ name="NYNJA"},
    #'Organization'{ name="Catalx"},
    #'Organization'{ name="FiaTech"},
    #'Organization'{ name="3Stars"},
    #'Organization'{ name="SwissEMX"},
    #'Organization'{ name="HistoricalPark"},
    #'Organization'{ name="Intralinks"} ],

    Structure =
```

```
  [ {"/erp/group", GroupOrgs},
    {"/erp/partners", PartnersOrgs},
    {"/erp/quanterall", HeadBranches} ],

    lists:foreach(fun({Feed, Data}) ->
       case kvs:get(writer, Feed) of
            {ok,_} -> skip;
         {error,_} -> lists:map(fun(X) ->
                      kvs:append(X,Feed)
       end, Data) end end, Structure).
```

<h4>PAY BOOT — Звітність CashFlow</h4>

Управління звітністю аутсорс підприємства доволі просте: 1) ми приймаємо оплати по інвойсам, виставленим клієнтам, з періодом раз на місяць; 2) ми виплачуємо зарплати раз на місяць. Тому згортки групуються календарно та зіпуються помісячно.

```
sal_boot() ->
   lists:map(fun(#'Product'{code=C} = P) ->
      lists:map(fun(#'Payment'{}=Pay) ->
         kvs:append(Pay,
         "/plm/"++C++"/outcome") end,
           salaries(C))
      end, products()).

pay_boot() ->
   lists:map(fun(#'Product'{code=C} = P) ->
      lists:map(fun(#'Payment'{}=Pay) ->
         kvs:append(Pay,
         "/plm/"++C++"/income") end,
           payments(C))
      end, products()).
```

<h4>PLM BOOT — Бюджетування проектів</h4>

Ініціалізація погодинна для кожного співробітника по проекту. Цей список буде використовуватись в майбутньому для розподілення виплат по опціонах.

```
assignees() ->
   lists:map(fun(#'Product'{code=C} = P) ->
      case kvs:get(writer,"/plm/"++C++"/staff") of
           {error,_} ->
             lists:map(fun(#'Person'{}=Person) ->
             kvs:append(Person,
             "/plm/"++C++"/staff") end,staff(C));
           {ok,_} -> skip end
      end, products()).
```

Виплати за процентами на субконто попроектно:

```
accounts() ->
  lists:map(fun(#'Product'{code=C}) ->
    lists:map(fun(#'Acc'{id=Id, rate=R}=SubAcc) ->
      Address = lists:concat(["/fin/acc/",C]),
      kvs:append(SubAcc,Address),
      Feed = lists:concat(["/fin/tx/",Id]),
      case kvs:get(writer, Feed) of
```

```
          {error,_} ->
            lists:map(fun(#'Payment'{
              invoice=I,price=P, volume=V}=Pay) ->
            kvs:append(rate(Pay,SubAcc,C), Feed)
            end, payments(C));
            {ok,_} -> skip
      end
    end, acc(C))
  end, plm_boot:products()).
```

За работу з даними відповідає біблиотека KVS, як працювати з нею читайте в минулих випусках журналу: — <a href="https://tonpa.guru/st 04-13— <a href="https://github.com/synrc/kvs»synrc/kvs </section> <section>

3.3 Інкапсуляція структури підприємства

Увесь код, який потрібен для створення фідів, зазвичай виносять в додаток з назвою ERP. Для кожного конкретного підприємства ми використовуємо свою Github організацію, можна навіть інше ім'я репозиторію, але ім'я Erlang/OTP додатку — завжди залишається ERP:

— <a href="https://github.com/enterprizing/erp»enterprizing/erp

У цьому репозиторії завжди будуть якісь реальні дані якоїсь компанії, яку ми автоматизуємо, якщо вони дозволять нам опублікувати свою організаційну структуру, первинні дані та словники. В інших випадках цей репозиторій робимо приватним. Самі додатки здатні працювати з довільними структурами ERP. </section> <section>

3.4 Приклади запитів до сховища

Elixir прелюдія:

```
defmodule PLM.Mixfile do
  use Mix.Project
  def project() do
    [
      app: :plm,
      version: "0.7.1",
      elixir: "~> 1.8.1",
      description: "PLM Product Lifecycle Management",
      deps: [{:bpe, "~> 4.7.3"}, {:erp, "~> 0.7.6"}]
    ]
  end
  def application(),
    do: [mod: {PLM.Application, []},
         applications: [:rocksdb, :kvs, :bpe, :erp]]
end
```

Зміст кореневої директорії БД підприємства:

```
>  :writer |> :kvs.all |> :lists.sort
[
  {:writer, '/acc/quanterall/Plovdiv', 3, [], [], []},
  {:writer, '/acc/quanterall/Sophia', 9, [], [], []},
  {:writer, '/acc/quanterall/Varna', 23, [], [], []},
  {:writer, '/bpe/hist/1562855060639704000', 1, [], [], []},
  {:writer, '/bpe/proc', 1, [], [], []},
  {:writer, '/erp/group', 1, [], [], []},
  {:writer, '/erp/partners', 7, [], [], []},
  {:writer, '/erp/quanterall', 3, [], [], []},
  {:writer, '/fin/acc/CATALX', 4, [], [], []},
  {:writer, '/fin/acc/NYNJA', 4, [], [], []},
  {:writer, '/fin/tx/CATALX/R&#38;D', 12, [], [], []},
  {:writer, '/fin/tx/CATALX/insurance', 12, [], [], []},
  {:writer, '/fin/tx/CATALX/options', 12, [], [], []},
  {:writer, '/fin/tx/CATALX/reserved', 12, [], [], []},
  {:writer, '/fin/tx/NYNJA/R&#38;D', 5, [], [], []},
  {:writer, '/fin/tx/NYNJA/insurance', 5, [], [], []},
  {:writer, '/fin/tx/NYNJA/options', 5, [], [], []},
  {:writer, '/fin/tx/NYNJA/reserved', 5, [], [], []},
  {:writer, '/plm/CATALX/income', 12, [], [], []},
  {:writer, '/plm/CATALX/investments', 4, [], [], []},
  {:writer, '/plm/CATALX/outcome', 12, [], [], []},
  {:writer, '/plm/CATALX/staff', 2, [], [], []},
  {:writer, '/plm/NYNJA/income', 5, [], [], []},
  {:writer, '/plm/NYNJA/investments', 2, [], [], []},
  {:writer, '/plm/NYNJA/outcome', 5, [], [], []},
  {:writer, '/plm/NYNJA/staff', 4, [], [], []},
  {:writer, '/plm/products', 2, [], [], []}
]
```

Список компаній, які входять в групу підприємства:

```
> :kvs.feed '/erp/group'
[{:Organization, 'Quanterall', 'quanterall.com', [], []}]
```

Список бренч-офісів головної (та єдиної) компанії групи:

```
> :kvs.feed '/erp/quanterall'
[
  {:Branch, '1562329445378242000',
   {:Loc, '1562329445378243000', [], 'BG', 'Plovdiv', [], []}},
  {:Branch, '1562329445378241000',
   {:Loc, '1562329445378242000', [], 'BG', 'Sophia', [], []}},
  {:Branch, '1562329445378234000',
   {:Loc, '1562329445378240000', [], 'BG', 'Varna', [], []}}
]
```

Список контрагентнів:

```
> :kvs.feed '/erp/partners'
[
  {:Organization, 'Catalx Exchange Inc.', 'catalx.io', [], []},
  {:Organization, 'HistoricalPark', [], [], []},
  {:Organization, 'NYNJA, Inc.', 'nynja.io', [], []},
  {:Organization, 'Intralinks', [], [], []},
  {:Organization, 'SwissEMX', [], [], []},
  {:Organization, 'FiaTech', [], [], []},
  {:Organization, '3Stars', [], [], []}
```

```
]
```

Бюджетування проекту по статтям субконто:

```
> :kvs.feed '/fin/acc/NYNJA'
[
  {:Acc, 'NYNJA/insurance', {2, 70}},
  {:Acc, 'NYNJA/reserved', {2, 10}},
  {:Acc, 'NYNJA/options', {2, 10}},
  {:Acc, 'NYNJA/R&#38;D', {2, 10}}
]
```

Виплати по опціонам для програмістів:

```
> :kvs.feed '/fin/tx/CATALX/options'
[
  {:Payment, '1562868880497278000', {0, 1}, {2, 150000}, 'USD', :crypto, [],
  {:Payment, '1562868880496849000', {0, 1}, {2, 100000}, 'USD', :crypto, [],
  {:Payment, '1562868880496409000', {0, 1}, {2, 120000}, 'USD', :crypto, [],
  {:Payment, '1562868880495897000', {0, 1}, {2, 150000}, 'USD', :crypto, [],
  {:Payment, '1562868880495412000', {0, 1}, {2, 100000}, 'USD', :crypto, [],
  {:Payment, '1562868880494920000', {0, 1}, {2, 100000}, 'USD', :crypto, [],
  {:Payment, '1562868880494538000', {0, 1}, {2, 100000}, 'USD', :crypto, [],
  {:Payment, '1562868880494072000', {0, 1}, {2, 50000}, 'USD', :crypto, [], []
  {:Payment, '1562868880493672000', {0, 1}, {2, 70000}, 'USD', :crypto, [], []
  {:Payment, '1562868880493387000', {0, 1}, {2, 150000}, 'USD', :crypto, [],
  {:Payment, '1562868880492939000', {0, 1}, {2, 120000}, 'USD', :crypto, [],
  {:Payment, '1562868880492234000', {0, 1}, {2, 120000}, 'USD', :crypto, [],
]
```

Люди, які працюють на проекті:

```
> :kvs.feed '/plm/NYNJA/staff'
[
  {:Person, '1562868880467887000', 'Maxim Sokhatsky', [], [], [], 1, []},
  {:Person, '1562868880467886000', 'Yuri Maslovsky', [], [], [], 8, []},
  {:Person, '1562868880467885000', 'Nikolay Dimitrov', [], [], [], 4, []},
  {:Person, '1562868880467884000', 'Radostin Dimitrov', [], [], [], 4, []},
  {:Person, '1562868880467883000', 'Georgi Spasov', [], [], [], 8, []}
]
```