ERP/1

Формальна модель та програмна архітектура
функціонального верифікованого мовного забезпечення
для побудови інфраструктурних процесінгових систем
орієнтованих на державну модель управління процесами,
зовнішнім аудитом, різними видами розподілених сховищ,
телекомунікаційними та реєстровими фреймворками,
інтернет-утворюючими сервісами зокрема та для
автоматизації захищених автономних офісів і
державних підприємств України у цілому.

Навчальний посібник курсу «Інформаційні системи»

Максим Сохацький
18 лютого 2024, Київ, Україна

УДК 002
УДК 004.4, 004.6, 004.9

Присвячується всім державним
службовцям України

Система управління державними підприємствами ERP/1 визначає формальну специфікацію та її імплементацію для сучасних оптимізованих підприємств які вимагають сучасних засобів контролю операцій та цілістності даних.

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/1 в нашому захищеному середовищі зі своїм центром випуску ECC X.509 сертифікатів. У цій книзі ви знайдете класичну авторську монографію на тему архітектури та імплементації такої системи, побудованої на міжнародних та державних стандартах України:

RFC: 7363, 6350, 4180, 5126, 5652, 8567, 9006, 9011, 9019, 9159, 9100, 8323, 7815, 7228, 6455, 8927, 8259, 4627, 7493, 7159, 4227, 3288, 6025, 5911, 4120, 4122, 7363, 6537, 6940, 7890, 2251-2256, 6960, 5280, 1034-1035, 4033-4035.

ISO: 19510, 19514, 42010, 18033, 14888, 10118, 10116, 15946, 29146, 9075, 27001, 19464, 20922, 21823, 27402, 30161, 30165, 20452, 42010, 19501, 19505, 8824-8825.

NIST: 800-162.

ДСТУ: 28147, 15946, 9798, 4145, 319-422, 319-122.

Постійне посилання твору: https://axiosis.top/sep/
Видавець: Державний науково-дослідний інститут МВС України

ISBN — 978-617-8027-23-0

Підготовлено до друку на Подолі, м. Київ.

Зміст

Передмова автора

Присвята

В колофоні написано шо посібник присвячено всім держслужбовцям України, це означає шо вони, як частина соціо-інформаційної системи державного устрою є основними кінцевими користувачами системи і для них, майбутніх розробників, операторів, архітекторів, керівників, аналітиків написаний цей твір.

Ідея виникнення і історія розвитку системи

Ідея написати ERP/1 виникла в мене одразу після початку шляху підприємництва, в 2005 році, коли я вибирав між OCaml, Haskell та Erlang платформу на якій будувати модель, яку я хочу впроваджувати. Так сталося, що під мої критерії мотивації підійшла тільки Erlang платформа і я почав шлях спочатку з соціальної турецької мережі, потім ПриватБанк, потім месенджер NYNJA, потім аспірантура по AXIO/1, потім міністерські системи.

Частини системи, такі як клієнтські компоненти для TWAIN сканування та роботи з Word, Excel документами написані на C#/F# так як перший досвід роботи був пов'язаний з .NET. Вже 18 років ERP/1 обслуговує потреби замовників, а її ідеї розповсюдилась за цей час на виробничий простір багатьох технологій, таких як LiveView, HTMX, Turbo, від більш старших: Ocsigen, WebSharper, Lift, Nitrogen.

Зараз ERP/1 на Elixir включає повністю всі рівні державних систем, а коштовність володіння як основний показник якості зменшений до мінімуму, так як вся система може бути запущена на одній ноді в одній віртуальній машині і написана на одній мові. Кожен з модулів ERP/1 вміщається на одну дискету 3.5" розміром 2.88 МБ.

Такі продуктові компанії як WhatsApp, ПриватБанк, RabbitMQ, Basho, AdRoll, Crytek Warface довели ефективність використання і низьку коштовність володіння, до прикладу компанія WhatsApp в момент продажу Facebook управлялась 6 інженерами і коштувала 19 млрд. доларів. Системи на базі ERP/1 побудовані з використанням цієї технології.

Розкриття компонент основної мотиваці

При розробці системи застосовувалася філософія мінімалізму в галузі програмного забезпечення[1], яку я розказував на одному з київських форумів в 2013 році. Якшо стисло, то вона зводиться до оптимізаційних критеріїв по часу, якості, людям і ресурсам: 1) зменшення часу виконання і розробки та збільшення часу життя системи і її ефективності; Основні показники: Computation/Time, Weeks/Release, Feature/Hour, Hour/Lifetime; 2) зменшення складності і ціни та збільшення надійності та простоти; Основні показники: Bugs/Code, Failures/Period, Cost/Support, Time/Fix; 3) зменшення невизначеності та збільшення зрозумілості і очікуваності; Основні показники: Commiters/App, Issues/Feature, Msgs/Issue, Commits/User; 4) зменшення обслуговування і додаткових ресурсів та збільшення ємності даних і обчислень; Основні показники: Cost/Information, Machines/User, Data/User, Size/Features, Requests/Time, Information/User.

Якшо звести ці всі показники до найголовнішої максими — то вона буде такою: ідеальний код це той, який найлегше верифікується (наочно, автоматично, математично), а це головним веде до його мінімальності. Наприклад, повна формалізація рівня Г7 по Т3І неможлива для таких систем як SQL, бо неможливо довести математично всі властивості пропрієтарних систем таких як Oracle, або навіть відкритих але складних систем як PostgreSQL. Тому при доведенні збереження властивостей системи використовуються безпосердній контроль колонок в базах даних і вручно пишуться всі рекурсори і фолди, як це роблять курсорні білдери фолдів в SQL. Тим не менш у складі Ericsson/OTP є реляційна база Mnesia, яка хоча теж містить всі недоліки складності, обмежена простим інтерфейсом в System F.

Коментарі та настанови для слухачів курсу

Цей курс є особливим в тому сенсі, що предмет дослідження, інфформаційна система ERP/1 одночасно добре формалізована і підходить у якості педагогічного дидактичного матеріалу, а також має стабільну історію впроваджень і підтримку сумісності.

[1] https://slides.com/maximsokhatsky/minimal/

Курс розділений на 12 частин, кожна з яких відповідає за певний етап розробки програмного забезпечення згідно ISO-9001, сюди входять: аналіз предметної області, робота з юридичними документами, формування архітектури і ескізного проекту, модель OSI і Закмана, аж до перевірки кваліфікаційного електронного підпису і шифрування по ДСТУ-4145 без допоміжних бібліотек. Цей посібник міг би бути також корисний для курсу по державному управлінню.

Цей посібник написаний з урахуванням вимог які висуваються для EDGE офісів, з підвищенним порогом входу по розміру коду, латенсі, відмовостійкості, розподіленості, вартості підтримки, ефективності, функціональності, відповідності до стандартів.

Подяки

Я би хотів подякувати всім вчителям, адже планетарно інститут педагогіки переживає кризу поваги до вчителів.

Розділ 1

Вступ

Цей посібник описує формальну модель та програмну архітектуру функціонального верифікованого мовного забезпечення для побудови інфраструктурних процесінгових систем орієнтованих на державну модель управління процесами для проведення зовнішнього аудиту, різними видами розподілених сховищ, телекомунікаційними та реєстровими фреймворками, інтернет-утворюючими сервісами зокрема та для автоматизації захищених автономних офісів і державних підприємств України у цілому.

Система управління державними підприємствами ERP/1 представлена у посібнику є не тільки ідіоматичним прикладом побудови інформаційних державних та комерційних систем у цілому, але і визначає формальну специфікацію та її імплементацію (з багатьма національними впровадженнями) для сучасних оптимізованих підприємств які вимагають сучасних засобів контролю операцій та цілістності даних.

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 управління підприємствами та інші інформаційні системи 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/1 в нашому захищеному середовищі зі своїм центром випуску ECC X.509 сертифікатів. У цій книзі ви знайдете перелік модулів системи та основні сутності схеми.

Універсальна платформа для створення та забезпечення функціонування інформаційних реєстрів баз (банків) даних різних масштабів: від базових міжсистемних довідників та класифікаторів, до високонавантажених корпоративних, місцевих та державних ресурсів. Цей посібник буде корисний всім, хто хоче зрозуміти які інформаційні системи застосовуються і державному і комерційному секторах.

Скорочення

ЄРЗ (Єдиний реєстр зброї НПУ), СУСЗЦЗ (Система управління силами та засобами цивільного захисту ДСНС), ЄІС (Єдина інформаційна система МВС), ЕСОЗ (Електронна система охорони здоров'я МОЗ), ФП МТРЗ (Функціональна підсистема матеріально-технічного та ресурсного забезпечення МВС), НГУ (Національна гвардія України), ДСНС (Державна служба з надзвичайних ситуацій МВС), ГСЦ (Головний сервісний центр МВС).

Розділ 2

Національна програма інформатизації

Мета Національної програми інформатизації (НПІ) — створення цифрового простору як проміжний птап розвитку інформаційного суспільства, прозорого правового середовища, яке захищене і базується на міжнародних стандртах та забезпечує інформаційні потреб та реалізацує права і свободи громадян на основі своєчасної, достовірної та повної інформації, підвищення ефективності державного управління.

Національна програма інформатизації (НПІ) веде свою історію з 74/98-ВР документа 1998 року, якій містив 28 статей, до поточного документа 2024 року 2807-IX, який містить вже 15 статей. Головним чином НПІ визначає протоколи запуску та термінації програм які містять наступні функції: експертизи, аналізу, формування, контролю, виконання, звітування програм інформатизації на державному (галузеві програми) ті місцевому (самоврядування) рівні, а також визначає суб єктів інформатизації: генеральний замовник — Міністерство цифрової трансформації, Керівник — посадова особа, виконавці та підрядники. НПІ є власником і розробником системи обліку таких програм. НПІ визначає процеси розробки згідно ISO/IEC 12207 та ISO 9001, а супроводу та підтримки згідно ISO/IEC/IEEE 14764:2022.

Основне завдання НПІ — безперевна оптимізація підприємств в структурі органів виконавчої влади, для цього система управління державою повинно визначати протоколи інституційної само-трансформації, а також визначати узгоджену архітектуру програмних систем поміж низки критичних міністерств.

2.1 Загальні принципи

На мета рівні неперервний процес реформування ОВВ зараз уявляється мені, як такий що керується наступними правилами: 1) Ін єктивність управління юстиції (як необхідний атрибут

кожного міністерства, аудит, розслідування); 2) IT-департамент або управління (у якості зовнішнього ЄДРПО, холдер продуктів Міністерства); 3) Управління трансформації (яка механізує процес IT-продуктами IT-департаменту). Далі управління Міністерства додаються в залежності від функцій Міністерства, але ці три плюс патронатна служба — обов'язкові. Міністерство Юстиції, Судова система, генеральна прокуратура, Поліція та інші агенції, як НАЗК, НАБУ, мають безпосередній або опосередкований доступ до (1). Це має регулюється відповідними ABAC правилами. Мінцифра має доступ до (3), як координатор мета-процесу трансформації. Також Мінцифра координує роботу і взаємодіє з (2) кожного міністерства для забезпечення каналу до реєстрів відповідних міністерств, які підтримуются відповідними IT-управліннями кожного мінстерства. Шини всіх документобігів і реєстрів координуються IT-управлінням Мінцифри, «Дія».

Завдання тансформації передбачає виконання (у тому числі) наступних цілей: 1) Кожне міністерство буде мати свій автономний і потужний IT-департамент, який обслуговує реєстри міністерства; 2) Міністерства будуть мати свої управління трансформації, працюватимуть на одному продукті, в якому будуть моделювати свою роботу; 3) Мінцифра як координатор буде затверджувати регламенти роботи IT-управлінь і управлінь трансформації кожного міністерства. «Дія» буде мати не тільки шину документообігу, продукт «Дія: Документообіг» (база), але і видавати ліцензії для учасників ринку (зараз 30 ліцензій).

2.2 Модель

1. Кабінет міністрів України
2. Міністерство освіти і науки України (МОН)
3. Міністерство охорони здоров'я України (МОЗ)
4. Міністерство внутрішніх справ України (МВС)
5. Міністерство закордонних справ України (МЗС)
5. Міністерство оборони України (МО)
6. Міністерство юстиції України (Мінюст)
7. Міністерство фінансів України (Мінфін)
8. Міністерство економіки України (Мінекономіки)
9. Міністерство енергетики України (Міненерго)
10. Міністерство соціальної політики України (Мінсоцполітики)
11. Міністерство регіональної політики України (Мінрегіон)
12. Міністерство цифрової трансформації України (Мінцифра)
13. Міністерство молоді та спорту України (Мінспорту)
14. Міністерство природних ресурсів України (Мінекології)
15. Міністерство аграрної політики України (Мінагрополітики)

2.3 Структурне ядро

1) Управління юстиції; 2) Департамент інформаційних систем виробничо-промислового управління; 3) Управління трансформації

Оскільки соціо-інформаційні системи повинні бути автономними та мати керований життєвий цикл, технічне відображення організаційної структури повинно бути під контролем міністерства, можливо у вигляді окремого підприємства (для убезпечення інтелктуальних ресурсів підприємства від ручного керування, а також оскільки IT-депратаменти міністерст відповідають за реєстри і персональні дані громадян).

Для керування структурою в реальному часі пропонується окремий вид протолу ОВВ 2.0 як наступне розширення після НПА до вже існуючого базового протоколу Документообігу згідно постанови №55 керуючого органу Кабінету Міністрів України. Наприклад, ДІТ (Державна Інституційна Трансформація).

В процесі як операційного документообігу (породжуваного структурними підрозділами міністерств), так і інституційного (породжуваного управліннями або агенціями державної цифрової трансформації) породжуються зліпки кваліфікованих електронних підписів посадових осіб які розслідуються як внутрішніми органами (відділи аудиту і відділи внутрішніх розслідувань), так і координуючим органом — Міністерством юстиції України.

2.4 Протокол державної інституційної транфсормації

Цей протокол визначає наступні операції над організаційною структурою і її політикою:

1) Створення і ліквідація структурних підрозділів; 2) Погодження та модифікація установчих конституційних документів; 3) Розробка технопроектів структурних підрозділів та їх систем; 3) Вибір і впровадження інформаційних систем для структурних підрозділів; 4) Запуск та операційна діяльність структурних підрозділів (виробництво);

Розділ 3

Органи виконавчої влади

Цей розділ описує структуру органів виконавчої влади (ОВВ), які є об'єктами інформатизації.

3.1 Міністерство науки і освіти

Ця секція є статтею-дослідженням таксономії структури Міністерства освіти і науки з точки зору як інформаційної автоматизованої системи так і соціальної структури з точки зору державного управління. Як приклад, в статті наводиться конкретна структура, яка є незначною модифікацією існуючої ієрархічної системи Міністерства освіти і науки.

3.1.1 Структурні підрозділи

1. Патронатна служба
2. Управління початкової школи
3. Управління середньої школи
4. Управління вищої школи
5. Управління юстиції
   - Департамент експертизи і сертифікації
   - Інститут інтелектуальної власності
   - Департамент кадрового забезпечення
   - Департамент аудиту і внутрішніх розслідувань
   - Департамент архівної справи (SCAN)
   - Технічний департамент (CA)
   - Департамент соціального і гуманітарного забезпечення
   - Відділ кадрів (ACC)
   - Юридичний департамент
   - Департамент міжнародного співробітництва
6. Управління науково-дослідними інститутами (агенція)
7. Національна академія наук (агенція)
   - Інститут формальної математики
     - Ректорат формальної філософії

    - Ректорат чистої математики
    - Ректорат прикладної математики
    - Ректорат мовного забезпечення
    - Ректорат теоретичної інформатики
  - Інститут формальної літератури
  - Інститут музики, кіно і образотворчого мистецтва
    - Національна консерваторія
    - Національна академія мистецтв
    - Національна кінематика
  - Інститут фізики і матеріалів
  - Інститут геології і геохімії
  - Інститут хімії і біології
  - Інститут соціальних і гуманітарних наук
    - Ректорат філософії
    - Ректорат археології
    - Ректорат національної історії
    - Ректорат права
    - Національна бібліотека
8. Управління політиками і структурними підрозділами (агенція)
  - Департамент комунікації (відділ кадрів)
  - Департамент контролю виконання показників (CRM)
  - Департамент планування переходу (аналіз процесів BPMN)
  - Департамент трансформації (широкий спектр спеціалізацій)

3.2 Міністерство охорони здоров'я

3.3 Міністерство внутрішніх справ

3.4 Міністерство закордонних справ

3.5 Міністерство оборони

Ця секція є дослідженням таксономії структури Міністерства оборони з точки зору як інформаційної автоматизованої системи так і соціальної структури з точки зору державного управління. Як приклад, в статті наводиться конкретна структура, яка є незначною модифікацією існуючої ієрархічної системи Міністерства оборони України, підсилена повним спектром інституцій для організації неперервного науково-освітнього і технологічно-виробничого процесу існування державного органу виконавчої влади — Міністерства оборони України.

У якості моделі гранулярності використана українська державна модель (міністерство, управління, департамент, відділ, сектор). Оскільки дана робота зосерджена в першу чергу на логіці існування процесу, тут значною мірою надається перевага формальним моделям, які потребують мінімальних зусиль для верифікації, моделювання і прогнозування. Оскільки формалізація процесу безпосередньо торкається інформаційного програмного забезпечення на користь приходять міжнародні стардарти телекомунікаційних протоколів, сертифікація яких торкається (в свою чергу) університетів, науково-дослідних інститутів, науково-виробничих інститутів. Науково-виробничі інститути використовується в широкому смислі як ті, що можуть бути комерційними угруповуваннями, міжнародними фундаціями, тощо. Для підтримки автономної діяльності ці всі інституції повинні входити в арсенал функціональних можливостей міністерства, включаючи головним чином університет четветого рівня акредитації, навчальні програми якого представлені частково обраними курсами п яти кафедр інституту математики НАН (див. Додатку 1). Розміщення інформаційної структури розгалуженої національної структури міністерства і його частин передбачає автономне забезпечення класу EDGE офіс з власним ресурсами охолодження, водо-електро-постачання, силами та засобами оборони.

Інформаційна політика формального моделювання передбачає довільне використання мовних сучаних засобів здатних до формальної верифікації (наявність промислових верифікаторів) ТЗІ рівня Г7 (повна математична верифікація) покладаючись основним чином на телекомунікаційні протоколи і міжнародні ISO стандарти (див. Додаток 5).

3.5.1 Мотивація

Основна мотивація даної роботи полягає у висвітленні таксономії Міністерства оборони України з точки зору оптимізації, автономності існування (sustainability), само-відтворюваності, підтримки життєвого циклу існування Міністерства.

3.5.2 Модель

1. Патронатна служба
2. Управління освіти і науки (агенція)
   — Університет четвертого рівня акредитації (див. Додаток 1)
   — Науково-дослідні інститути
   — Науково-виробничі інститути
3. Управління медицини (агенція)
   — Клінічні наукові дослідження та лабораторії, НДІ ПВМ (MED)
   — Реабілітації («Пуща-Водиця», «Трускавецький», «Хмельник»)
   — Клінічні, мобільні (4) лікарні, госпіталі (14)
   — Медичні служба (5 родів), тактична медицина, медичні сили
   — Інститут медицини (ЗДМУ, ХНМУ, ЛНМУ, ТНМУ)
4. Виробничо-промислове управління (агенція)
   — КБ (ДАТ «Укроборонпром», ДАХК «Артем»)
   — Департамент економічного моделювання і планування
   — Департамент ресурсного забезпечення (SCM, TMS, WMS)
   — Департамент бюджетування і закупівель (FIN)
   — Департамент будівництва і архітектури, ліній виробництва
   — Телекомунікаційний департамент інформаційних систем
      — Відділ систем врядування
      — Відділ облікових систем
      — Відділ телекомунікаційних систем
      — Відділ безпекових протоколів Інтернет
   — Департамент авіації, авіоніки, аеронавтики і безпілотних літаючих апаратів і їх систем
      — Відділ авіації
      — Відділ авіоніки
      — Відділ аеронавтики
      — Відділ безпілотних систем
   — Департамент машинобудування (terrain) (SolidWorks)
   — Департамент кораблебудування
5. Управління юстиції
   — Відділ експертиз і сертифікації (ISO/IETF)
   — Департамент архівної справи (SCAN)
   — Департамент аудиту і внутрішніх розслідувань
   — Технічний департамент (CA)
   — Департамент соціального і гуманітарного забезпечення
   — Відділ кадрів (ACC)
   — Юридичний департамент
   — Департамент міжнародного співробітництва
6. Управління розвідки
7. Головне управління позиційною політикою
   — Мобілізаційний департамент
   — Департамент навчальних програм
   — Департамент сил і засобів оборони CRM (див. Додаток 2)

    - Родина сухопутних військ
    - Родина повітряних сил
    - Родина воєнно-морських сил
    - Родина спеціальних сил (ССО, МС, РЕБ, РХБЗ, ТРО)
    - Родина кібербезпеки і ДШВ
  - Департамент контролю і управління DFR (див. Додаток 4)
  - Економічний департамент
    - Відділ ресурсного забезпечення (WMS)
    - Відділ бюджетування і закупівель
8. Управління політиками і структурними підрозділами (агенція)
  - Департамент комунікації (відділ кадрів)
  - Департамент контролю виконання показників (CRM)
  - Департамент планування переходу (аналіз процесів BPMN)
  - Департамент трансформації (широкий спектр спеціалізацій)

3.5.3 Принципи

Одним з головних принципів закладених у фундамент МО є принцип вищої освіти, яка здобувається згідно до вимог міжнародних стандартів. Для забезпечення потреб існування працівників всіх сфер цієї таксономії в основу її неперервної маніфестації покладено існування університету четвертого рівня акредитації з усіма спеціальностями необхідними для покриття потреб самого МО.

Одним з універсальних принципів закладених в фундамент МО є принцип розподілу влади, з якого випливає три корпуси МО: 1) Політичний корпус (головне управління, перехідне управління, патронатна служба), 2) Виконавчий корпус (Управління освіти і науки, Медичне управління, Виробничо-промислове управління), 3) Судовий корпус (Управління юстиції, трибунал). Політичний корпус передбачає виділення окремої агенції яка здійснює запуск процесів створення і апробація структурних підрозділів міністерства. Сюди також входить структурний підрозділ — головне управління яке управляє силами (ЗСУ) та засобами оборони. Виконавчий корпус убезпечує інтелектуально-ємні структурні підрозділи і виокремлює їх під автономний контроль агенції з більшою дотичністю до зовнішніх структур. Судовий корпус — окремий стурктурний підрозділ з процесами аудиту і внутрішніх розслідувань в існуючих соціально-інформаційних системах в структурі МО.

3.5.4 Структурні підрозділи

1. Розподіл влади і мінімізація таксономії
2. Нормалізація формальних процесів
3. Аудит міністерства і процес трансформації
4. Архітектура структурних підрозділів
5. Апробація результатів і циклічність мета-процесу

Розподіл влади є головним принципом ефективного управління. Контролюючі органи повинні спеціалізуватися на розслідуваннях і бути виокремлені. Політики процесів і реквізитної інформації повинні здійснюватися політичним органом. Політика повинна здійснюватися окремими виконавчими органами (освіта, наука, виробництво).

Вимоги до документування процесів повинні бути на найвищому рівні (еквіваріантні семантики, спрощення процесів до нормальних форм, мінімазія горизонтальних зв язків), відповідати вимогам постанови №55 КМУ і наказу №124 МО. Повинна бути організований процес історіографії і аудиту діло-процесів згідно стандарту ISO-19510 для аудиту NATO. Запуск діло-виробництва передбачається поступово і гранулярно, спочатку від головних і малоресурних проєктів управління політик і далі згідно стратегії управління по іншим структурним підрозділам.

3.5.4.1 Патронатна служба

Сприяння реалізації політичних цілей Міністра, консультування Міністра, організаційне, інформаційне, експертно-аналітичне забезпечення діяльності Міністра.

3.5.4.2 Управління освіти і науки (агенція)

Науковий процес крім освітнього видіяє науково-дослідний і наково-виробничий у тих сферах, яких потребують департаменти організаційної структури МО. Ці компанії повинні знаходитися як і університет у сфері впливу МО.

3.5.4.3 Виробничо-промислове управління (агенція)

Пропонується повне дублювання сфер виробництва засобів оборони у відповідні департаменти, так як це є основними ресурсами головного управління тому доцільно зберігати бачення повної картини під ієрархією МО. Існуючі виробничі процеси зосереджені в «Укроборонпром» і «Артем», пропонується розглядати як зовнішні конструкторські бюро, в одному переліку з комерційними підприємствами які працюють можливо навіть проектно.

3.5.4.4 Управління юстиції

Управління юстиції бере на себе функції обліку сил (відділ кадрів) та протокольних дій в середині системи, які аналізуються антикорупційним відділом департаменту аудиту і внутрішніх розслідувань. Тут також зосереджені юридичні функції які обслуговують всю ієрархію МО, а також зовнішніх партнерів (NATO). Також тут зосереджений центр видачі криптографічних ключів всієї ієрархії.

3.5.4.5 Головне управління

Спрощені функції головного штабу з локальним резервом сил та засобів оборони. Головна консоль (Godot) управління підпорядковує логіці ведення позиційної політики силами та засобами оборони з прогнозування подій та ціною їх усунення. Позиційна стратегія передбачає локальний облік сил та засобів оборони.

3.5.4.6 Управління політиками і структурними підрозділами (агенція)

Формальна модель адміністративного управління переходу (або врядування) від існуючої структури до будь-якої наперед заданої (як приклад наведеної). Перехідне управління займається аудитом існуючих процесів (юридичне забезпечення) та їх трансформації в урядові та облікові системи міністерства у взаємодії з Телекомунікаційним департаментом, контролем їх виконання: впровадження, апробації, калібрування, проектний менеджмент.

Процес трансформації розділений на наступні категорії (обернено до швидкоплинності): 1) трансформація (діджіталізація) існуючих процесів без модифікації логіки, такі як діловодство; 2) створення нових процесів і технологічного забезпечення для вакантних департаментів і відділів, такі як виробничо-промислове управління; 3) планування і розвиток вакантних департаментів і управлінь (довготривалий процес, розвиток навчальних програм, побудова та підтримка життєвого циклу продуктів).

Кожна фаза процесу трансормації передбачає побудову життєвого циклу проєкту, а також продукту, який покладений в його основу, сюди входить повний набір документації: Технічне Завдання, Ескізний проєкт, Технічний проєкт, Технічна документація. Управління політиками поділяє сфери компетенції з Телекомунікаційним департаментом виробничо-промислового відділу і загальні правила документування.

Важливі, головні процеси управління політиками, як основні його функції можна класифікувати так: 1) архівна справа процесного виробництва; 2) розробка процесів запуску структурних підрозділів; 3) розробка процесів документообігу; 4) розроб-

ка процесів управління і координації; 5) розробка технологічних процесів ISO-9001.

3.5.5 Результати

В статті представлена запропонована таксономія нормалізованого міністерства з урахування міжнародного досвіду організації державних структур з чітким і прозорим принципом розподілу влади і організації глобального (стратегічного) і локального (операційного) контекстів для виконання своїх функції у найпростіший і безпосередній спосіб мінімізуючи кількість протоколів взаємодії всередині системи.

В додатках статті представлена таксономія освітніх програм університету четвертого рівня акредатації і таксономія телекомунікаційного депратемнту, у функції якого входять розробка і впровадження системи. В процесі виконання завдання первинного моделювання було здійснено намагання охопити ключові органи, аж до рівня гранулярності відділів і секторів (побудова первинного індексу) і їх горизонтальних протоколів взаємодії.

3.5.6 Бібліографія

ДСТУ 2732-94 Діловодство і архівна справа.
26.05.2014 №333 Інструкція з обліку особового складу.
21.11.2017 №608 Порядок проведення службового розслідування.
26.07.2018 №370 Інструкція з діловодства.
07.04.2017 №124 Інструкція з діловодства.
07.10.2015 №393 Положення Про Юридичну Службу.
29.11.2018 №604 Інструкція з надання доповідей і донесень про події, кримінальні правопорушення, військові адміністративні правопорушення та адміністративні правопорушення, пов'язані з корупцією, порушення військової дисципліни та їх облік.

Розділ 4

Специфікація та сертифікація

4.1 Законодавча база

4.1.1 Загальні положення

Базова версія «MIA: Документообіг» керується наступними загальними положеннями які виражені законами України:

— 2657-XII, Про інформацію[1],
— 7498-ВР, Про Національну програму інформатизації[2],
— 39396-ВР, Про звернення громадян[3],
— 2939-VI, Про доступ до публічної інформації[4]
— 2155-VIII, Про електронні довірчі послуги[5],
— 851-IV, Про електронні документи та електронний документообіг[6],

та розпорядженнями і постановами Кабінету Міністів України:

— 386-2013-р, Розпорядження КМУ #3860-Р [7],
— 373-2006-п, Постанова КМУ #373 [8].

[1] https://zakon.rada.gov.ua/laws/show/2657-XII
[2] https://zakon.rada.gov.ua/laws/show/74/98-вр
[3] https://zakon.rada.gov.ua/laws/show/393/96-вр
[4] https://zakon.rada.gov.ua/laws/show/2939-17
[5] https://zakon.rada.gov.ua/laws/show/2155-19
[6] https://zakon.rada.gov.ua/laws/show/851-15
[7] https://zakon.rada.gov.ua/laws/show/386-2013-р
[8] https://zakon.rada.gov.ua/laws/show/373-2006-п

4.1.2 Базова версія «MIA: Документообіг»

Продукт «MIA: Документообіг» в основному базується на Поставі #55 Кабінету Міністрів України та інших постановах КМУ:

— 55-2018-п, КМУ. Постанова #55 Деякі питання документування управлінської діяльності [9],
— 749-2018-п, КМУ. Постанова #749 Про затвердження Порядку використання електронних довірчих послуг в органах державної влади, органах місцевого самоврядування, підприємствах, установах та організаціях державної форми власності[10],
— v0144774-20, ДП «Український науково-дослідний і навчальний центр проблем стандартизації, сертифікації та якості". Наказ #144 Про прийняття та скасування національних стандартів ДСТУ 4163:2020 та ДСТУ 9031:2020[11],

але платформа продукту базується на наказаї Міністерства юстиції України, Міністерства цифрової трансформації, Міністерства освіти і науки, та Законами України:

— z1854-12, Міністерство юстиції України. Наказ #16005 Про затвердження Порядку роботи з електронними документами через систему електронної взаємодії органів виконавчої влади з використанням електронного цифрового підпису[12],
— z1039-20, Міністерство цифрової трансформації України. Адміністрація державної служюи спеціального зв'язку та захисту інформації України. Наказ #140614[13],
— z1306-11, Міністерство освіти і науки, молоді та спорту України. Наказ #1207 Про вимоги до форматів даних електронного документообігу в органах державної влади. Формат електронного повідомлення[14],
— z1421-14, Міністерство юстиції України. Наказ #1886/5 Про затвердження Порядку роботи з електронними документами у діловодстві та їх підготовки до передавання на архівне зберігання[15], — 851-IV, Про електронні документи та електронний документообіг[16],
— 8094-ВР, Про захист інформації в інформаційно-телекомунікаційних системах[17],

---

[9] https://zakon.rada.gov.ua/laws/show/55-2018-п
[10] https://zakon.rada.gov.ua/laws/show/749-2018-п
[11] https://zakon.rada.gov.ua/rada/show/v0144774-20
[12] https://zakon.rada.gov.ua/laws/show/z1854-12
[13] https://zakon.rada.gov.ua/laws/show/z1039-20
[14] https://zakon.rada.gov.ua/laws/show/z1306-11
[15] https://zakon.rada.gov.ua/laws/show/z1421-14
[16] https://zakon.rada.gov.ua/laws/show/851-15
[17] https://zakon.rada.gov.ua/laws/show/80/94-вр

4.1.3 Розширення та додаткові модулі

Розширення «MIA: Провадження»

— 4651-VI, Кримінальний процесуальний кодекс України[18],
— v0298905-20, Офіс генерального прокурора. Наказ #298 Про затвердження Положення про Єдиний реєстр досудових розслідувань, порядок його формування та ведення[19],

Розширення «MIA: Закупівлі»

— 922-19, Закон України про публічні закупівлі[20], — 169, Постанова КМУ #169,
— 1178 Постанова КМУ #1178,
— 808-20 Закон України про оборонні закупівлі,
— 1275-2022-п Постанова КМУ #1275,
— 1070-2019-п Постанова КМУ #1070,
— 224-2020-п Постанова КМУ #224,
— 710-2016-п Постанова КМУ #710,
— z0500-20 Наказ Мінекономіки #708,
— 544-2016-п Постанова КМУ #544,
— v1749731-15 Наказ Мінекономіки #1749,
— 1495-п Постанова КМУ #1495.

Розширення «MIA: Зброя»

— 5708, Проект Закону про право на цивільну вогнепальну зброю[21], — 5709, Проект Закону про внесення змін до Кодексу України про адміністративні правопорушення та Кримінального кодексу України для реалізації положень Закону України "Про право на цивільну вогнепальну зброю"[22],

[18] https://zakon.rada.gov.ua/laws/show/4651-17
[19] https://zakon.rada.gov.ua/laws/show/v0298905-20
[20] https://zakon.rada.gov.ua/laws/show/922-19
[21] http://w1.c1.rada.gov.ua/pls/zweb2/webproc4_2pf3516=5708skl=10
[22] http://w1.c1.rada.gov.ua/pls/zweb2/webproc4_2pf3516=5709skl=10

Розширення «MD: Військова частина»

— ДСТУ 2732-94, Діловодство і архівна справа.
— 26.05.2014 #333, Інструкція з ведення обліку особового складу.
— #608 Порядок проведення службового розслідування.
— 26.07.2018 #370, Інструкція з діловодства.
— 07.04.2017 #124, Інструкція з діловодства.
— 07.10.2015 #393, Положення Про Юридичну Службу.
— 29.11.2018 #604, Інструкція з надання доповідей і донесень про події, кримінальні правопорушення, військові адміністративні правопорушення та адміністративні правопорушення, пов'язані з корупцією, порушення військової дисципліни та їх облік.

4.2 Класифікація вимог

4.2.1 Вимоги до інтерфейсу користувача

4.2.2 Вимоги до адміністрування системи

4.2.3 Вимоги типових ділопроцесів системи

4.2.4 Вимоги процесінгової системи

4.2.5 Вимоги інтеграції з зовнішніми системами

4.2.6 Вимоги до розподіленої роботи

4.2.7 Вимоги до комплексу засобів захисту (КЗЗ)

4.2.8 Технічні вимоги до зберігання даних

4.3 Відповідність міжнародним стандартам

4.3.1 Стандарти RFC

4.3.2 Стандарти ISO

4.3.3 Національні стандарти ДСТУ та NIST

4.4 Засоби захисту та ступені гарантії безпеки

4.4.1 Мануальна наочна верифікація

4.4.2 Інтеграційне тестування

4.4.3 Математична верифікація

Розділ 5

Державна система

По аналогії зі стандартом ISO 42010 «Фреймворку Закмана», фреймворк Максима Сохацького визначає та уточнює архітектурні рівні з яких складаються сучасні корпоративні інформаційні системи:

— Юридично-документальний рівень
— Обліково-реєстровий рівень
— Зв'язність людей та пристроїв
— Генерація, валідація і верифікація
— Телекомунікаційна платформа і безпека інтернету

5.1 Юридично-документальний рівень

Згідно фреймворку верхній шостий рівень визначає BPMN процеси згідно яких здійнюється відзеркалення юридично-правивих відносин електронного документообігу. Кожен крок такого процесу, та усі його документи підписуються особистим ключем КЕП посадової особи, що дає змогу проведення диспутів та розслідувань Міністерством юстиції України. Окрім того цей рівень системи орієнтований на аналітику у взаємодії з громадянами через СЕВ ОВВ.

Юридично-документальні системи ERP/1 будуються на сховищі з єдиним простором ключів Facebook RocksDB, що здатне працювати через Intel SPDK на NVMe дисках, наприклад у складі таких сховищ як CEPH. Обсяг обігу документів на великих підприємствах сягає 1ТБ на рік.

5.2 Обліково-реєстровий рівень

Обліково-реєстровий рівень пропонує низькорівневе масштабоване розподілене журнальне сховище даних та метаданих, яке може бути побудоване на реляційних базах даних, базах даних з єдиним простором ключів з гарантіями консистентності (chain-hash) або їх комбінаціях.

Класичні представники цього рівня в системах управління підприємствами: система управління людськими та матеріальними ресурсами, банківські системи PCI DSS, складські системи, медичні інформаційні системи, системи управління поставками та виробництвом, системи сервісних послуг, системи управління проектами, тощо.

5.3 Технологічний рівень зв'язності людей та пристроїв

5.3.1 Локальний

Рівень зв'язності людей та пристроїв визначає комунікаційні протоколи та технології, які об'єднують головні ресурси підприємства (пристрої та людей) у одну телекомунікаційну мережу. Як правило виробництво складається з багатьох пристроїв що підключаються до промислових шин як MQTT, та робочих місць користувачів, каналів зв'язку з інформаційними системами, корпоративні та національні шини, тощо.

Цей рівень також визначає засоби мастшабування пам'яті (персистентної та волатильної) та обчислювальних ресурсів (за допомогою процесінгових брокерів доставки повідомлень). Це рівень визначає реляційні бази даних та бази даних з єдиним простором ключів, а також стандарти та протоколи передачі інформації у промислових ERP системах, такі як CSV, JSON, SOAP, BERT, ASN.1, тощо.

5.3.2 Крос-системний

Крім того Мінцифра підтримує два середовища інтеграційної взаємодії національного рівня, учасниками яких є суб'єкти господарювання індексовані ЄДРПО підприємства:

1) національна система електронної взаємодії органів виконавчої влади (СЕВ ОВВ) з відкритим ринком клієнтів[1] і широким охопленням органів виконавчої влади[2]. Ця шина діє на рівні юридично-документального рівня і безпосередньо пов'язана з серверами документообігу, які є учасниками цієї взаємоії;

2) національна система електронної взаємодії державних електронних інформаційних ресурсів[3] (СЕВДЕІР «Трембіта»), яка представляє собою спеціалізовану версію Ubuntu з пакетами X-ROAD, власною інфраструктурою CA, TSP, OCSP серверами і шифрованими каналами передачі конфіденційної інформації.

[1] https://se.diia.gov.ua/sedlist — Перелік сертифікованих систем документообігу

[2] https://se.diia.gov.ua/uploads/documents/45.xlsx — Перелік систем документообігу і їх ЄДРПО підключених до національної шини СЕВ ОВВ

[3] https://catalog.trembita.gov.ua/?env=SEVDEIR — Каталог сервісів «Трембіта»

5.4 Генерація, валідація і верифікація

Рівень схеми даних визначає модель зберігання даних як з точки зору об'єктів-сутностей так і з точки зору технологій та протоколів, які необхідні для їх опису. Головним чином це Фреймворк Закмана та сімейство стандартів які описують UML, System F та необхідні генератори SDK, верифікатори типів (валідатори), моделі процесів, тощо.

5.5 Безпека інтернету та інфраструктури

Рівень безпеки визначає схему функціонування основного центрального засвідчувального орнагу, акредитованих центрів сертифікації ключів, протоколи шифрування та підпису, директорію підприємства, інтернет протоколи найменування ресурсів, шифровані протоколи комунікації, диспетчерські системи трафіку повітряних суден, тощо. Усе визначено згідно ASN.1 специфікації і стандартів протоколів серії X[4].

[4] https://www.itu.int/itu-t/recommendations/index.aspx?ser=X

Розділ 6

Юридично-документальний рівень

6.1 Вступ

Згідно фреймворку системи верхній п'ятий рівень визначає BPMN процеси згідно яких здійнюється віддзеркалення юридичних відносин у вигляді електронного документообігу. Кожен крок такого процесу, та усі його документи підписуються особистим ключем КЕП посадової особи, що дає змогу проведення диспутів та розслідувань Міністерством юстиції України з власним Засвідчувальним органом простором інтернет імен. Окрім того цей рівень системи орієнтований на аналітику у взаємодії з громадянами через СЕВ ОВВ.

Юридично-документальні системи ERP/1 будуються на сховищі з єдиним простором ключів Facebook RocksDB, що здатне працювати через Intel SPDK на NVMe дисках, наприклад у складі таких сховищ як CEPH. Обсяг обігу документів на великих підприємствах сягає 1ТБ на рік.

6.1.1 Види документообігів

6.1.1.1 Постанова 55 КМУ

6.1.1.2 Наказ 124 МО

6.1.1.3 Закупівлі

6.1.1.4 Провадження

6.1.1.5 Зброя

6.1.2 Функціональні можливості

6.2 Модулі підприємства

ERP/1 є комплексом бібліотек (N2O.DEV) та підсистем додатків (ERP.UNO), який використовує загальну шину і загальну розподілену базу даних для швидкіснх операційних вітрин.

ERP — Даний модуль обліково-реєстраційного рівня зберігає основну ієрархічну структуру підприємства, її схему, метаінформацію про типи даних, а також сам інформіцію: записи про персонал, інвентар, компанії та офіси підприємства.

CRM — Система управління зв'язками з громадськістю та органами виконавчої влади: являє собою базову реалізацію постанови #55 КМУ.

CART — Система управління реєстрами: являє собою реалізацію базового сервера для реєстрових задач.

6.3 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.

6.4 Архітектура врядувальних CRM систем

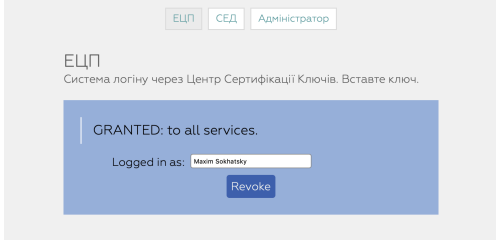
6.4.1 Сторінки

Перелік сторінок

```
def route(<<"ldap", _::binary>>), do: LDAP.Index
def route(<<"crm",  _::binary>>), do: CRM.Index
def route(<<"rmk",  _::binary>>), do: RMK.Index
def route(<<"kvs",  _::binary>>), do: KVS.Index
def route(<<"act",  _::binary>>), do: BPE.Actor
def route(<<"help", _::binary>>), do: HELP.Index
```

6.4.1.1 LDAP

Сторінка авторизації користувачів.



Сторінка авторизації

6.4.2 Комболукап

6.4.3 Сервіси

6.4.4 CEB OBB

6.4.5 Шаблони

6.4.6 Дерева

6.4.7 Процеси

Даний модуль інкапсулює визначення Схеми, Бізнес-Процесів та Форм, які використовуються у системі Infotech-ERP згідно методології фреймворку Захмана.

6.4.7.1 Формування нормативно-довідкової інформації

Виділяються наступні основні процеси організаційно-розпорядчих документів: «Накази», «Протоколи», «Доручення керівництва».

6.4.7.2 Обробка вхідних документів

Вхідні документи надходять в УДСД, ВОРЗГ та ВОДПІ. При надходженні документа уповноважена посадова особа зазначених СП реєструє його в системі та виконується його подальша обробка.

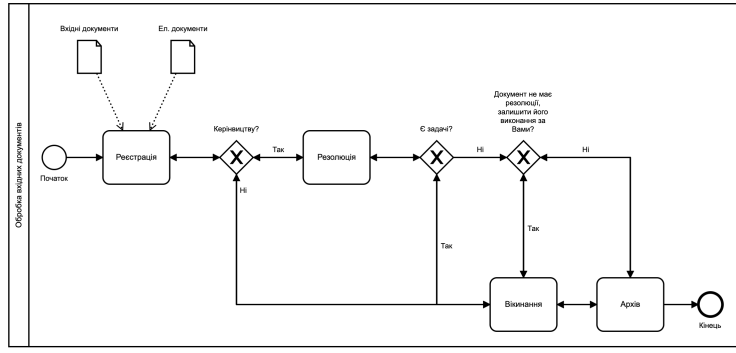


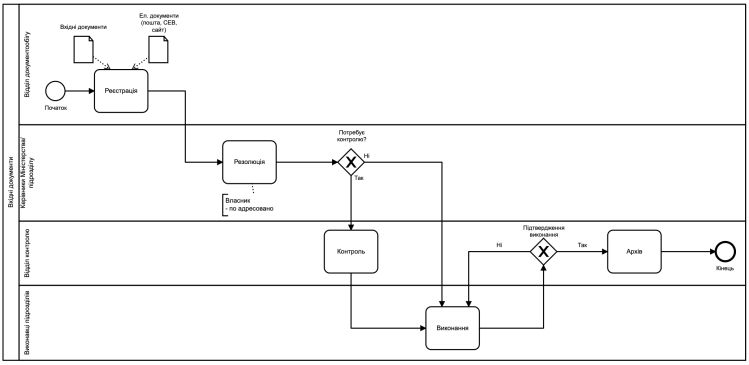
Рис. 6.2 Бізнес-процес обробки вхідних докумнетів

Далі вхідний документ надходить або Міністру/Заст. Міністра/-Державному секретарю для накладання резолюції, або до СП на виконання.

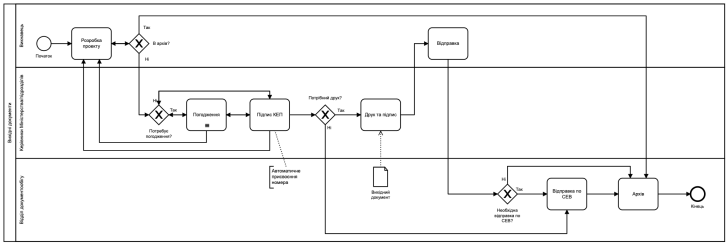
6.4.7.3 Вхідні документи

6.4.7.4 Вихідні документи

Вихідні документи створюються в підрозділах (ініціатор документа). Вихідні документи можуть виникати з ініціативи співробітників Міністерства або ж в результаті обробки вхідних документів. Якщо вихідний документ пов'язаний з вхідним документом, то відповідальному працівнику необхідно вказати посилання на пов'язаний документ. Обробка документів виконується по одному бізнес-процесу, незалежно від місця виникнення документа.



Бізнес-процес вхідних докумнетів



Бізнес-процес вихідних докумнетів

В системі реєструється проект вихідного документа, який повинен бути погоджений з переліком погоджуючих осіб. Після підпису фінальним підписантом, документу присвоюється номер та виконується відправка.

6.4.7.5 Внутрішні документи

Внутрішні документи можуть вводитися всіма учасниками документообігу. Виділяються наступні основні бізнес-процеси внутрішніх документів: «Доповідна записка», «Лист».

6.4.7.6 Організаційно-розпорядні документи

Розробка проекту документа

На даному етапі розробляється електронний проект документа: заповнюються всі необхідні реквізити в електронній картці документа, після збереження електронної картки автоматично прикріплюється шаблон документа як оригінал. Виконавець вносить вміст документа в оригінал і зберігає його. Ініціатор додає всіх виконавців, кому адресований наказ. По полю «Адресовано» ав-

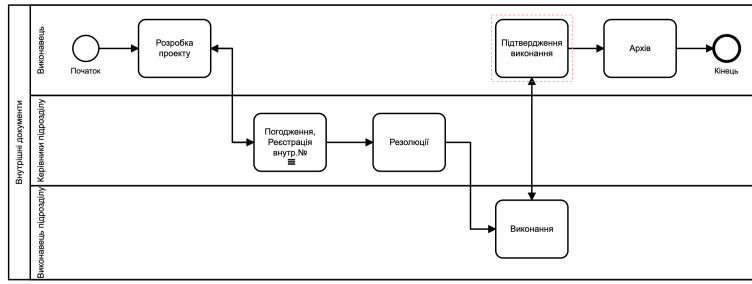


Рис. 6.5 Бізнес-процес внутрішніх докумнетів

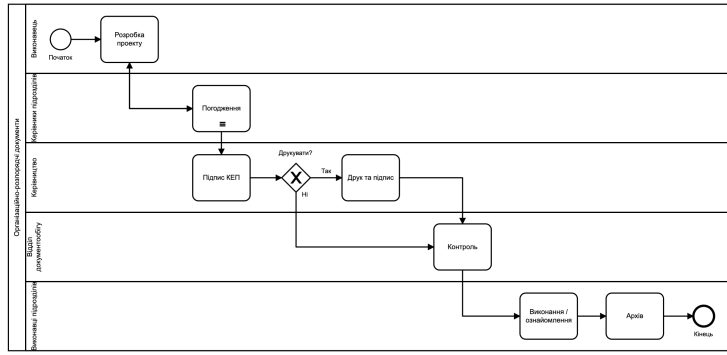


Рис. 6.6 Бізнес-процес організаційно-розпорядних докумнетів

томатично будуть створені задачі на виконавців. Далі необхідно передати документ на наступний крок.

Погодження

На даному кроці виконується погодження документа особами, які були вказані Виконавцем при створенні проекту документа. Обов'язковою умовою передачі документа на наступний етап - позитивне погодження від ВСІХ погожуючих осіб. Інакше далі передати документ неможливо. Якщо один з візуючих відхилив документ (при цьому вноситься коментар з причинами відхилення і зауваженнями до документа) - в даному випадку документ повертається на першу стадію Виконавцю на доопрацювання. Якщо всі особи погодили документ - він автоматично передається на наступну стадію. Після погодження документу можна сформувати Аркуш погодження у вигляді друкованої форми.

КЕП

На даній стадії документ підписується в електронному вигляді Керівництвом Міністерства. Під час цього документу присвоюється реєстраційний номер. У разі налагодження СЕД на використання QR-коду, він розміром 21 на 21 мм розміщується в нижньому лівому куті першої сторінки документа. У разі налагодження СЕД на використання штрих-коду, він розміщується у правому кутку нижнього поля першої сторінки документа.

Підпис

Після підписання організаційно-розпорядчого документу, у разі необхідності створення паперового варіанту уповноважена особа служби (помічник) Міністра/заступника міністра/державного секретаря роздруковує документ та надає на підпис керівнику. Якщо організаційно-розпорядчийдокумент підписано керівником підрозділу, то при необхідності документ роздруковує виконавець.

Постановка на контроль

На даному етапі контролюючий СП перевіряє завдання по документу, при необхідності здійснює постановку на контроль, та періодичність. Документи і задачі на контролі незалежно від кроку опрацювання документа доступні за окремим фільтром, їх можна відстежувати незалежно від того, на якій стадії знаходиться документ, контролювати виконання, проводити аналіз, і т.д.

Виконання/Ознайомлення

На даному етапі контролюючий СП перевіряє завдання по документу, при необхідності здійснює постановку на контроль, та періодичність. Документи і задачі на контролі незалежно від кроку опрацювання документа доступні за окремим фільтром, їх можна відстежувати незалежно від того, на якій стадії знаходиться документ, контролювати виконання, проводити аналіз, і т.д.

Цифровий шифровий архів

Після ознайомлення документ переходить в Архів, який додатково накладає підписи КЕП Архіву та утримує істрорію всіх проміжних сертифікатів АЦСК до ЦЗО..

6.4.7.7 Звернення громадян

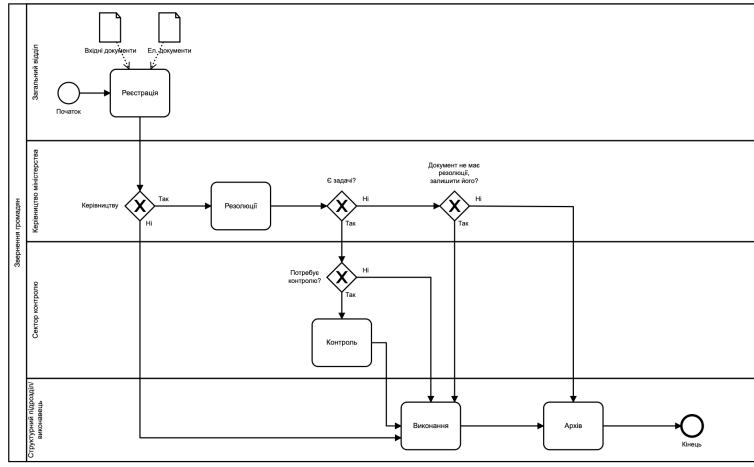


Рис. 6.7 Бізнес-процес звернення громадян

6.4.8 Елементи

Тут зібрана мінімальна кількість бізнес-форм, специфічних для CRM СЕД, яка необхідня для забезпечення реалізації функціональних вимог замовника.

6.4.8.1 Календар

Календар взятий з бібліотеки NITRO, проте потребує додаткової стилізації.

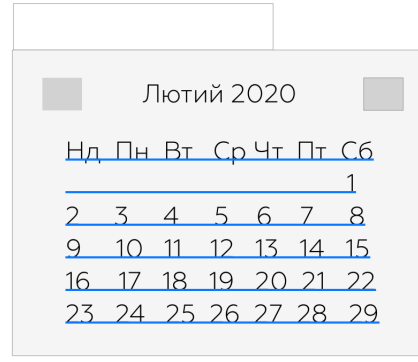
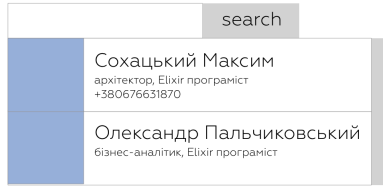


Рис. 6.8 Контрольний елемент Календар

6.4.8.2 Пошук по довільним фідам

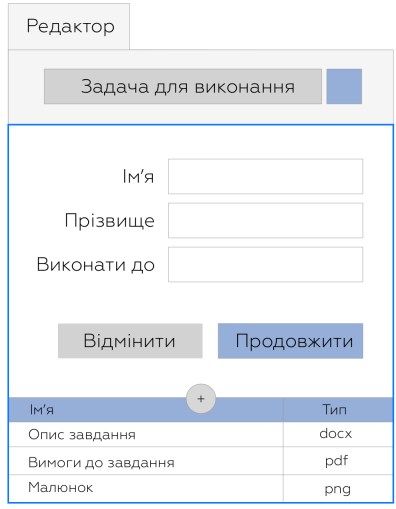
Для забезпечення пошуку по словникам та бізнес-об'єктами системе передбачається створення спеціалізованого скалярного комбо-пошуку по довільним фідам в сховищі даних. Наприклад: Співробітники, Населені пункти КОАТУУ, тощо.



Контрольний елмент віддаленого пошуку по базі даних Рис. 6.9

6.4.8.3 Форма редагування та пошуку

Для кожного типу документу в системі реєструються дві форми: форма пошуку та форма редагування (вона ж форма створення нового). Наявність двох форм вмотивована відмінністю валідаторі: для пошуку валідатори повинні дозволяти пусті поля, позаяк для редагування валідатори повинні перевіряти валідність полів бізнес-об'єктів.



Контрольний елемент редагування документу та підлеглих файлів Рис. 6.10

6.4.8.4 Управління бізнес процесом

Для управління завдання, доступу до документів процесу, створення нових документів в процесі, візування, підпису, проштовху документів по бізнес-процесу використовується стандартний контрольний елемент управління бізнес-процесом.

Рис. 6.11 Контрольний елемент управління бізнес-процесами

6.4.8.5 Документи в бізнес-процесах

При навігації по дукументам процесу передбачається миттєве відображення підлеглого документа в лівій панелі головної сторінки користувацького інтерфейсу.

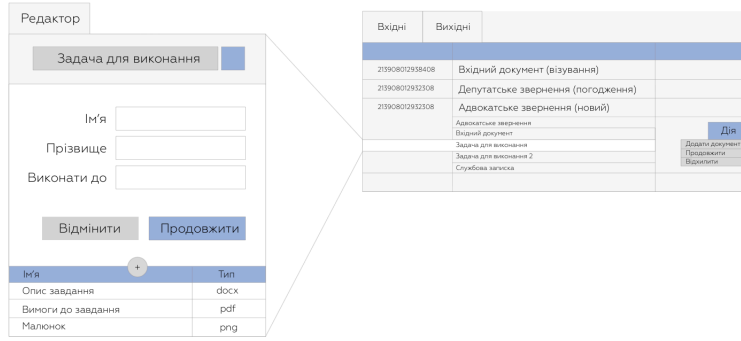
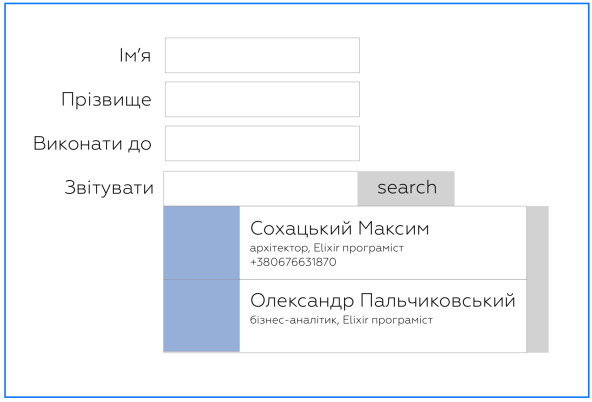


Рис. 6.12 Навігація по документам бізнес-процесу

6.4.8.6 Використання контролів на формах

Приклад використання контрольного елементу довільного пошуку на формах.



Приклад використання контрольних елементів на формах Рис. 6.13

6.4.8.7 Контрольний елемент КОАТУУ

Приклад використання контрольного елементу КОАТУУ.

Приклад використання контрольного елементу КОАТУУ Рис. 6.14

6.4.9 Редактори

Тут будуть перелічені контроллери сторінок, кожна з яких є SPA веб додатком.

6.4.9.1 Вхід в систему

Сторінка входу в систему з використанням ЕЦП.

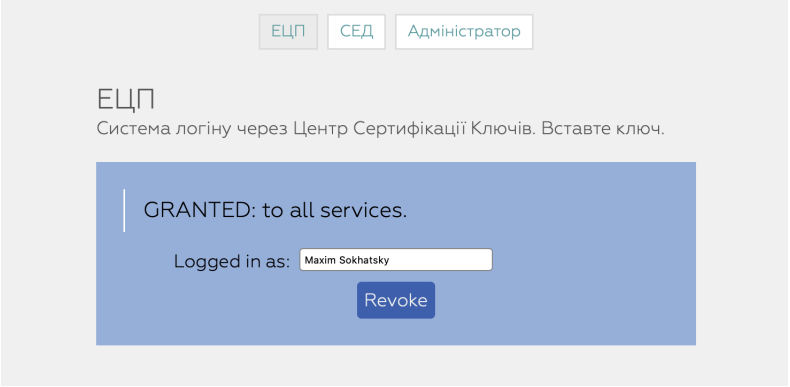
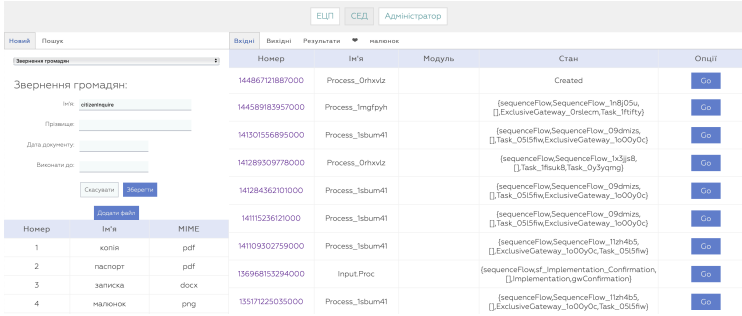


Рис. 6.15 Сторінка входу в систему

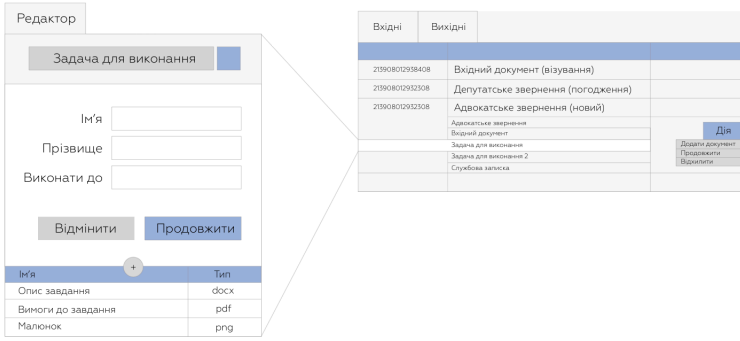
6.4.9.2 Робота з документами

Головна сторінка системи для роботи з документами в бізнес-процесах.



Сторінка роботи з документами

При навігації по дукументам процесу передбачається миттєве відображення підлеглого документа в лівій панелі головної сторінки користувацького інтерфейсу.



Навігація по підлеглим документам

Р

6.4.10 Конструктор

Тут представлені адміністративні сторінки управління системою.

6.4.10.1 Бізнес-об'єкти

Глобальний каталог усіх бізнес-об'єктів системи.

6.4.10.2 Бізнес-процеси

Перелік усіх зареєстрованих бізнес-процесів в системі, та можливість їх тестування.

6.4.10.3 Бізнес-форми

Перелік усіх форм документів та бізнес-форм користувача, зареєстрованих в системі.

6.4.11 Мова програмування FormalTalk

Розділ 7

Обліково-реєстраційний рівень

7.1 Вступ

Обліково-реєстровий рівень пропонує низькорівневе масштабоване розподілене журнальне сховище даних та метаданих, яке може бути побудоване на реляційних базах даних, базах даних з єдиним простором ключів з гарантіями консистентності (chain-hash) або їх комбінаціях. Класичні представники цього рівня в системах управління підприємствами: система управління людськими та матеріальними ресурсами, банківські системи PCI DSS, складські системи, системи управління поставками та виробництвом, системи сервісних послуг, системи управління проектами, тощо.

7.1.1 Види реєстрів

1) Реєстри орієнтовані на суб'єктів організаційних систем;
2) Реєстри орієнтовані на облік матеріальних ресурсів;
3) Реєстри орієнтовані на георафічні об'єкти;
4) Реєстри орієнтовані на події;
5) Реєстри орієнтовані на документи, накази, НПА;
5) Реєстри медичних систем (FHIR);
5) Реєстри предметно-орієнтованих словників для функціональних підсистем.

7.1.2 Функціональні можливості

7.2 Модулі підприємства

ERP/1 є комплексом бібліотек (N2O.DEV) та підсистем додатків (ERP.UNO), який використовує загальну шину і загальну розподілену базу даних для швидкіснх операційних вітрин.

FIN — Фінансовий модуль підприємства для бухгалтерії, 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 календарі.

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

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

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

WMS — Система управління складом, устаткуванням, деталями.

TMS — Система управління транспортом підприємства.

HL7 — Медична система, яка реалізує міжнародний FHIR стандарт.

7.3 Архітектура облікових CART систем

7.3.1 Облік метаінформації

7.3.2 Облік ABAC правил

7.3.3 Облік інфраструктури і само-моніторинг

7.3.4 Облік словників і класифікаторів

7.3.5 Адміністративний облік

7.3.6 Облік таксономії предметної облісті

7.3.7 Облік процесів предметної області

Розділ 8

Технологічний рівень зв'язності людей та пристроїв

8.1 Вступ

Ця глава визначає форамальну специфікацію на програмне забезпечення усіх рівнів моделі Закмана для підприємств ISO-42010, містить широкий спектр прикладів, розказує про складові компоненти та є вичерпним авторським стартовим посібником для курсу навчання розробки технологічних програм для платформи Erlang і публікації в системі електронної взаємодії державних електронних інформаційних ресурсів "Трембіта".

Трембіта — це система побудована на пакетах Ubuntu 18 LTS і пропонує головним чином інфраструктуру X.509, а також розгортання національного форку шини X-ROAD, який використовується як сереовище для гетерогенних сервісів, в якому всі 15 міністерств публікують свої сервіси і їх клієнтські адаптери. Каталог сервісів доступний публічно[1]. Сам інтерфейс управління X-ROAD написаний з використаннями SOAP/WSDL специфікацій. В цьому цифровому просторі відбувається взаємодія (передача конфіденційних даних в основному) між обліково-реєстраційними системами міністерств на основі безпосередніх захищеник каналів зв'язку між сервісами та їх споживачами, яка містить просту і фіксовану логіку.

СЕВ ОВВ, на відміну від системи Трембіта — це публічна шина даних для урядової кореспонденції і нормативно-правових актів, вона побудована згідно ISO/IEC 11756:2010 (MUMPS) на базі продукту InterSystems Caché. В цьому цифровому просторі відбувається робота систем врядування юридично-документального рівня

[1] https://catalog.trembita.gov.ua

8.4 Обчислювальні ресурси

Концептуальна модель системи в рамках якої функціонує N2O визначаена як обчислювальне середовище, яке складається з процесору подій (N2O), операційного (ETS) та персистентного сховища (KVS). З точки зору обчислювального середовища, ресурси підприємства складаються з глобального сховища та обчислень, які розділяють глобальну адресацію та представляють собою Erlang-процеси (N2O протоколи). Кожен процес PI, може містити певний набір протоколів, будь-який з яких відповідає на певний набір повідомлень. Протоколи N2O визначені на точці підключення повинні не перетинатися, в іншому випадку протокольні модулі можуть перехоплювати та впливати на інші протокольні модулі, які повинні реагувати на той самий тип повідомлень.

Усі асинхронні процеси PI запускаються під головним супервізором n2o та індексуються URI ключем разом з типом реактивного каналу реального часу: ws або mqtt. N2O протоколи підключені безпосередньо до веб-сокет точок підключення виконуються в контексті TCP процесів, у даному випадку TCP-сервера бібліотеки RANCH, супервізор ranch_sup.

```
> :supervisor.which_children :n2o
[
  { {:ws, '/chat/ws/4'}, <0.985.0>, :worker, [:n2o_ws] },
  { {:ws, '/chat/ws/3'}, <0.984.0>, :worker, [:n2o_ws] },
  { {:ws, '/chat/ws/2'}, <0.983.0>, :worker, [:n2o_ws] },
  { {:ws, '/chat/ws/1'}, <0.982.0>, :worker, [:n2o_ws] },
  { {:mqtt, '/bpe/mqtt/4'}, <0.977.0>, :worker, [:n2o_mqtt] },
  { {:mqtt, '/bpe/mqtt/3'}, <0.976.0>, :worker, [:n2o_mqtt] },
  { {:mqtt, '/bpe/mqtt/2'}, <0.975.0>, :worker, [:n2o_mqtt] },
  { {:mqtt, '/bpe/mqtt/1'}, <0.974.0>, :worker, [:n2o_mqtt] },
  { {:caching, 'timer'}, <0.969.0>, :worker, [:n2o] }
]
```

8.4.1 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.

8.5 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 повністью 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.

8.6 Середовище

Для забезпечення повного замкненого середовища пропонують наступні заміни бібліотек kernel та stdlib:

🚀 VM — віртуальна машина середовища виконання[2]
☆ BASE — базова системна бібліотека як заміна stdlib[3]
☆ RT — бібліотека середовища виконання як заміна kernel[4]
📨 SYN — бібліотека PubSub для розподілених систем[5]
⚡ MAD — бібліотека управління пакетами та інстансами[6]

[2] vm.n2o.dev
[3] base.n2o.dev
[4] rt.n2o.dev
[5] syn.n2o.dev
[6] mad.n2o.dev

8.6.1 Бібліотеки

Для запезпечення повноцінної промислової специфікації ERP/1, ми розширили набір інструментальних засобів наступними бібліотеками: формальними представленнями презентаційного рівня FORM та системою управління бізнес-процесів BPE. FORM представляє собою декларативну бібліотеку побудови графічних інтерфейсів, а бібліотека BPE підтримує XML файли стандарту BPMN 2.0 та реалізує безпосередню інтерналізацію BPMN семантики у семантику віртуальної машини Erlang.

- N2O — сервер протоколів для стандартів MQTT/WS/QUIC[7]
- NITRO — UI веб-фреймворк Nitrogen[8]
- KVS — бібліотека доступу до KV сховищ RocksDB[9]
- FORM — бібліотека декларативного конструювання іформ[10]
- BPE — сисема управління процесами стандарту BPMN 2.0[11]
- RPC — бібліотека генерації SDK для мов JS, protobuf, Swift[12]

8.6.2 Приклади

Головні приклади фундації N2O.DEV присвячені наступним темам: MQTT та WebSocket чати для демонстрації веб-фреймворку NITRO, який працює як модуль N2O, приклад REST адаптер до бази даних KVS, та повністю чистий N2O додаток CHAT на основі бібліотеки SYN без використання NITRO:

- SAMPLE — ідіоматичний приклад Nitrogen поверх WS[13]
- REVIEW — ідіоматичний приклад Nitrogen поверх MQTT[14]
- REST — бібліотека для побудови HTTP API[15]
- CHAT — приклад системи доставки повідомлень[16]

[7] ws.n2o.dev
[8] nitro.n2o.dev
[9] kvs.n2o.dev
[10] form.n2o.dev
[11] bpe.n2o.dev
[12] rpc.n2o.dev
[13] sample.n2o.dev
[14] review.n2o.dev
[15] rest.n2o.dev
[16] chat.n2o.dev

8.7 Протоколи, схеми та мови їх опису

8.7.1 Мова опису протоколів ASN.1

8.7.2 Мова опису протоколів SOAP/XSD/XML

8.7.3 JSON валідатори draft-07 і JTD

8.8 Формати передачі даних

8.8.1 Бінарні формати ETF/BERT

8.8.2 Бінарні формати DER/BER/PER

8.8.3 Колоночний текстовий формат CSV/CSM

8.8.4 Текстові формати JSON і XML

8.9 Розробка Інтернет додатків

8.9.1 Erlang та сучасний веб

Erlang реалізує недосяжну мрію кожного обчислювального середовища для паралельної та узгодженої конкуретної обробки повідоблень. Так найбільш відомі бібліотеки акторів (Akka, Orleans), які реалізують основні примітиви: процесори та черги, копіюють модель акторів Erlang, зазвичай намагаються також реалізують додатково механізми перезапуску та супервізії процесів подібно до Erlang, проте тільки Erlang забезпечує soft real-time характеристики, завдяки керуванню латенсі з точністю до таймінгу команд віртуальної машини. А з виходом 24 версії в 2020 році, яка почала підтримувати JIT-компіляцію завдяки asmjit, продуктивність та чуттєвість віртуальної машини зросла ще більше.

З формальної точки зору достатньо добре ізольоване середовище віртуальної машини Erlang не тільки забезпечує характерстики реального часу для SMP-планувальника легких зелених процесів, але і обмежує область видимості heap пам'яті виключно для процесів-власників, що унеможливлює вплив відмови певних процесів на глобальний стан віртуальної машини.

Erlang ідеально підходить для побудови високо-навантажених, просто-масштабованих, подійно-орієнтованих, неблокуючих, надійних, постійно-доступних, високо-ефективних, швидких, безпечних та надійних систем обробки повідомлень та розподілених у просторі та часі систем.

8.9.2 DSL vs Шаблони

З технічної точки зору N2O успішно показує неперевершену досі якість DSL програмування, яку ви не зможете знайти в сучасних веб-фреймворках для мов Erlang та Elixir. За 7 років неперервної еволюції N2O ми переписали кожен з 700 рядків по 30 разів, якшо порахувати через коміти Github. Веб-фреймворк NITRO, сховище KVS, та BERT.JS кодування може забезпечити відображення в веб-браузері повноекранних вертикальних форм з усіма обчислюваними полями зі швидкістю 60 форм в секунду по веб-сокет каналу. А надзвичайно компакта JavaScript бібліотека-компаньйон вміщується в 4 MSS/MTU вікна — саме такий розмір мінімального веб-клієнта з BERT кодуванням, який повністю управляється зі сторони сервера.

N2O сервер та веб-фреймворк NITRO реалізують концепцію не тільки управління сесіями та каналами, але і усім стеком побудови додатків включаючи UI частину, як це відбувається у таких веб-фреймворках як Erlang Nitrogen, OCaml Ocsigen, Scala Lift,

F# WebSharper, а завдяки таким розширенням як FORM та BPE ідеально підходять і для побудови автоматизованих CRM систем.

Це не означає, що за допомогою N2O ви не можете створювати більш класичні та архаїчні додатки у стилі DTL шаблонізаторів, або як це відбувається у таких фреймворках як PHP, ASP, JSP, Rails, тощо. Перші версії NITRO містили в прикладах використання Django Template Library (DTL), проте задля чистоти стеку були прийнято не включати в N2O додаткові шаблонізатори крім NITRO DSL.

8.9.3 Історія

N2O сервер, а також NITRO веб-фреймворк були спроектовані як інструментільні засоби для створення промислових ERP модулів підприємства у складі відкритої платформи ERP/1. Напочатку, N2O був відгалужений, як оптимізована версія веб-фреймворку Nitrogen, створеного Расті Клопгаузом. Хотілося оптимізувати та вдосконалити мінімізований WebSocket-тракт, який не містить синхронного протоколу HTTP взагалі та дозволяє створювати повноцінні асинхронні веб-додатки реального часу. На ньому була створена система управління депозитами в національному банку ПриватБанк. Пізніше N2O був розділений на бібліотеку-фреймворк процесів та протоколів (власне N2O) та бібліотеку-веб-фреймворк NITRO. Бібліотеки N2O та NITRO також отримали можливість роботи не тільки через WebSocket але і через MQTT та через чисті TCP або UDP. Така оновлена версія 5.10 була впроваджена як ядро системи повідомлень для додатку NYNJA з відкритим open-source протоколом і саме ій присвячений друга версія підручника.

8.9.4 Інтерфейс NITRO

8.9.5 Сховище KVS

8.9.6 Логіка BPMN

8.9.7 Додатки MQTT та WebSocket

Розділ 9

Генерація, валідація і верифікація

Розділ 10

Інфраструктурний рівень безпеки інтернету

10.1 Електронний підпис і цифрова печатка

Кваліфікований Електронний Підпис, або Кваліфіклована Електронна Печатка — це набір стандартів криптографічного захисту ДСТУ 4145, та міжнародних стандартів які визначають його конверт: X.501, X.509, X.511, X.520.

Серія міжнародних стандартів X.500, групується по категоріям, кожна з яких має свій перелік ASN.1 файлів. Аби підключити усі визначення необхідні для КЕП використані наступі компоненти стандартів (виділені болдом): X.501 — BasicAccessControl [1], InformationFramework [2], UsefulDefinitions [3]; X.509 — SpkmGssTokens [4], PkiPmiExternalDataTypes [5], AttributeCertificateDefinitions [6], AlgorithmObjectIdentifiers [7],
AuthenticationFramework [8], CertificateExtensions [9];
X.511 — SpkmGssTokens [10], DirectoryAbstractService [11]; X.520 — PasswordPolicy [12], UpperBounds [13], SelectedAttributeTypes [14].

Можно було би винести необхідні визначення одразу в KEP.asn1, однак цим хотілося підкреслити сумісність з міжнародними стандартами. Окрім серії протоколів X.500, КЕП ще визначає також запити та відповіді OCSP, також у ASN.1 форматі.

[1] https://www.itu.int/ITU-T/formal-language/itu-t/x/x501/2019/BasicAccessControl.html
[2] https://www.itu.int/ITU-T/formal-language/itu-t/x/x501/2019/InformationFramework.html
[3] https://www.itu.int/ITU-T/formal-language/itu-t/x/x501/2019/UsefulDefinitions.html
[4] https://www.itu.int/ITU-T/formal-language/itu-t/x/x509/2019/ExtensionAttributes.html
[5] https://www.itu.int/ITU-T/formal-language/itu-t/x/x509/2019/PkiPmiExternalDataTypes.html
[6] https://www.itu.int/ITU-T/formal-language/itu-t/x/x509/2019/AttributeCertificateDefinitions.html
[7] https://www.itu.int/ITU-T/formal-language/itu-t/x/x509/2019/AlgorithmObjectIdentifiers.html
[8] https://www.itu.int/ITU-T/formal-language/itu-t/x/x509/2019/AuthenticationFramework.html
[9] https://www.itu.int/ITU-T/formal-language/itu-t/x/x509/2019/CertificateExtensions.html
[10] https://www.itu.int/ITU-T/formal-language/itu-t/x/x511/2019/SpkmGssTokens.html
[11] https://www.itu.int/ITU-T/formal-language/itu-t/x/x511/2019/DirectoryAbstractService.html
[12] https://www.itu.int/ITU-T/formal-language/itu-t/x/x520/2019/PasswordPolicy.html
[13] https://www.itu.int/ITU-T/formal-language/itu-t/x/x520/2019/UpperBounds.html
[14] https://www.itu.int/ITU-T/formal-language/itu-t/x/x520/2019/SelectedAttributeTypes.html

На відміну від самого алгоритму КЕП, який визначено ДСТУ 4145, конверти визначаються не стандартами, а наказами міністерства юстиції: Проект наказу Адміністрації Держспецзв'язку та Держкомінфоматизації (2009) [15], Наказ Міністерства юстиції України 1236/5/453 [16]. Керуючись цими нормативними документами було створено файл KEP.asn1 [17], який є одним з трьох top-level файлів необхідниї для компіляції ASN.1 компілятором[18].

Існує небагато безкоштовних та повних компіляторів (генераторів парсерів) ASN.1 специфікацій. Erlang є прикладом системи, до складу якої входить першокласний безкоштовний з відкритою ліценцією ASN.1 компілятор, де файли в ASN.1 нотації можуть бути зкомпільовані безпосередньо Erlang компілятором:

```
> erlc AuthenticationFramework.asn1
> erlc InformationFramework.asn1
> erlc KEP.asn1
```

Створити файл підпису PKCS-7 можна за допомогою будь якої програми сертифікованої в Україні. Найпростіше отримати свою КЕП печатку будучи клієнтом ПриватБанку. За допомогою "Користувача ЦСК"компанії ІІТ ви можете підписувати файли використовуючи безкоштовну форму приватного ключа у вигляді звичайного файлу.

[15] http://www.dsszzi.gov.ua/dsszzi/control/uk/publish/article?art_id=77726
[16] https://zakon.rada.gov.ua/laws/show/z1401-12
[17] https://github.com/synrc/ca/blob/master/priv/kep/KEP.asn1
[18] https://asn1.erp.uno

10.1.1 Приклад використання

Щоб показати як користуватися КЕП, та прочитати атрибутивну інформацію з сертифікату, який вшитий в PCKS-7 повідомлення з криптографічним підписом, покажемо 5 функцій:

```
> CA.CAdES.readSignature
[
  {:certinfo, ~c"TINUA-2955020254",
   "СОХАЦЬКИЙ МАКСИМ ЕРОТЕЙОВИЧ",
   "МАКСИМ ЕРОТЕЙОВИЧ", "СОХАЦЬКИЙ",
   "СОХАЦЬКИЙ МАКСИМ ЕРОТЕЙОВИЧ",
   [
     subjectKeyIdentifier: "VNXfTvJQccGtPgNhUftIQZV+mUROTgroLotsbtYZsFE=",
     authorityKeyIdentifier: "XphNUm+C84/0vi5ABGgN/rOvysLkBHVNB9CuTISwfB0=",
     keyUsage: [<<6, 192>>],
     certificatePolicies: {"https://acsk.privatbank.ua/acskdoc",
      ["1.2.804.2.1.1.1.2.2", "1.3.6.1.5.5.7.2.1"]},
     basicConstraints: [],
     qcStatements: {"https://acsk.privatbank.ua",
      ["0.4.0.1862.1.1", "0.4.0.1862.1.5", "1.3.6.1.5.5.7.11.2",
       "0.4.0.194121.1.1", "1.2.804.2.1.1.1.2.1"]},
     cRLDistributionPoints: ["http://acsk.privatbank.ua/crl/PB-2023-S6.crl"],
     freshestCRL: ["http://acsk.privatbank.ua/crldelta/PB-Delta-2023-S6.crl"],
     authorityInfoAccess: [
       {"1.3.6.1.5.5.7.48.2",
        "http://acsk.privatbank.ua/arch/download/PB-2023.p7b"},
       {"1.3.6.1.5.5.7.48.1", "http://acsk.privatbank.ua/services/ocsp/"}
     ],
     subjectInfoAccess: [
       {"1.3.6.1.5.5.7.48.3", "http://acsk.privatbank.ua/services/tsp/"}
     ],
     subjectDirectoryAttributes: [
       {"1.2.804.2.1.1.1.11.1.4.7.1", "0"},
       {"1.2.804.2.1.1.1.11.1.4.1.1", "2955020254"}
     ]
   ], "ФІЗИЧНА ОСОБА", "", "", ~c"UA", "КИЇВ"},
  {:certinfo, ~c"UA-14360570-2310",
   "КНЕДП АЦСК АТ КБ ПРИВАТБАНК\"\"", "", "",
   "КНЕДП АЦСК АТ КБ ПРИВАТБАНК\"\"",
   [
     contentType: "0.6.9.42.840.113549.1.7.1",
     signingTime: "240221110356Z",
     messageDigest: "MfvlhoDVCPkptQRN+S2zNGp0nrOsS93mLdbcz/kZ9GI=",
     signingCertificateV2: 5400415814250126491315088041558718376138774192 68,
     contentTimestamp: {"1.2.840.113549.1.7.2",
      369952533463044024072847521187489 7026, "20240221110626Z",
      "MfvlhoDVCPkptQRN+S2zNGp0nrOsS93mLdbcz/kZ9GI="}
   ], "АТ КБ ПРИВАТБАНК\"\"", "", "", ~c"UA", "Київ"}
]
```

10.2 Криптографічні інформаційні повідомлення

Реалізації повинні підтримувати транспортування ключів, узгодження ключів і раніше розподілені симетричні ключі шифрування ключів, представлені ktri, kari та kekri відповідно.

Реалізації можуть підтримувати керування ключами на основі пароля, представлене pwri. Реалізації МОЖУТЬ підтримувати будь-які інші методи керування ключами, такі як шифрування на основі ідентифікації Боне-Франкліна та Боне-Бойєна (RFC 5409) або інші методи шифрування SYNRC, такі як варіанти KYBER Key Transport (LAMPS-WG) для постквантової криптографії (PQC).

IETF (SMIME-WG) стандарти: 5990, 5911, 5750–5754, 5652, 5408, 5409, 5275, 5126, 5035, 4853, 4490, 4262, 4134, 4056, 4010, 3850, 3851, 3852, 3854, 3855, 3657, 3560, 3565, 3537, 3394, 3369, 3370, 3274, 3114, 3278, 3218, 3211, 3217, 3183, 3185, 3125–3126, 3058, 2984, 2876, 2785, 2630, 2631, 2632 , 2633, 5083, 5084, 2634.

Сумісність: Erlang SSL, LibreSSL CMS, OpenSSL CMS, GnuPG S/MIME.

10.2.1 Головна функція

Специфікація синтаксису криптографічних повідомлень CMS X.509 для дисципліни RSA (Key Transport), ECC (Key Agreement), KEK (Key Encryption Key) для додатків Erlang/OTP, які ніколи не переживала heartbleed (!) CRYPTO та SSL. Реалізовано як модуль CMS програми CA.

```
defmodule CMS do
    def decrypt(cms, {schemeOID, privateKeyBin}) do
        {_,{:ContentInfo,_,{:EnvelopedData,_,_,x,y,_}}} = cms
        {:EncryptedContentInfo,_,{_,encOID,{_,<<_::16,iv::binary>>}},data} = y
              case :proplists.get_value(:kari,  x, []) do
        [] -> case :proplists.get_value(:ktri,  x, []) do
        [] -> case :proplists.get_value(:kekri, x, []) do
        [] -> case :proplists.get_value(:pwri,  x, []) do
        [] -> {:error, "Unknown Other Recepient Info"}
              pwri  -> pwri(pwri,   privateKeyBin, encOID, data, iv) end
              kekri -> kekri(kekri, privateKeyBin, encOID, data, iv) end
              ktri  -> ktri(ktri,   privateKeyBin, encOID, data, iv) end
              kari  -> kari(kari,   privateKeyBin, schemeOID, encOID, data, iv)
     end
    end
end
```

10.2.2 CMS-KARI-ECC

IETF 3278:2002 Використання алгоритмів криптографії еліптичних кривих (ECC) у синтаксисі криптографічних повідомлень (CMS) із підтримкою Suite B IETF 5008:2007, 6318:2011.

```
# openssl cms -decrypt -in encrypted.txt -inkey client.key -recip client.pem
# openssl cms -encrypt -aes256 -in message.txt -out encrypted.txt \
                       -recip client.pem -keyopt ecdh_kdf_md:sha256
```

CMS Codec KARI: ECC+KDF/ECB+AES/KW+256/CBC:

```
def map(:'dhSinglePass-stdDH-sha512kdf-scheme'), do: :sha512
def map(:'dhSinglePass-stdDH-sha384kdf-scheme'), do: :sha384
def map(:'dhSinglePass-stdDH-sha256kdf-scheme'), do: :sha256
def eccCMS(ukm, bit), do:
    :'CMSECCAlgs-2009-02'.encode(:'ECC-CMS-SharedInfo', sharedInfo(ukm, bit))
def sharedInfo(ukm, len), do: {:'ECC-CMS-SharedInfo',
    {:'KeyWrapAlgorithm',{2,16,840,1,101,3,4,1,45},:asn1_NOVALUE}, ukm, <>}

def kari(kari, privateKeyBin, schemeOID, encOID, data, iv) do
    {_,:v3,{_,{_,_,publicKey}},ukm,{_,kdfOID,_},[{_,_,encryptedKey}]} = kari
    {scheme,_}  = CA.ALG.lookup(schemeOID)
    {kdf,_}     = CA.ALG.lookup(kdfOID)
    {enc,_}     = CA.ALG.lookup(encOID)
    sharedKey   = :crypto.compute_key(:ecdh,publicKey,privateKeyBin,scheme)
    {_,payload} = eccCMS(ukm, 256)
    derived     = KDF.derive(map(kdf), sharedKey, 32, payload)
    unwrap      = CA.AES.KW.unwrap(encryptedKey, derived)
    res         = CA.AES.decrypt(enc, data, unwrap, iv)
    {:ok, res}
end
def testDecryptECC(), do: CA.CMS.decrypt(testECC(), testPrivateKeyECC())

def testECC() do
    {:ok,base} = :file.read_file "priv/certs/encrypted.txt"
    [_,s] = :string.split base, "\n\n"
    x = :base64.decode s
    :'CryptographicMessageSyntax-2010'.decode(:ContentInfo, x)
end

def testPrivateKeyECC() do
    privateKey = :public_key.pem_entry_decode(pem("priv/certs/client.key"))
    {:'ECPrivateKey',_,privateKeyBin,{:namedCurve,schemeOID},_,_} = privateKey
    {schemeOID,privateKeyBin}
end
```

10.2.3 CMS-KEKRI-KEK

Інформація про одержувача ключа шифрування ключа, як визначено CMS IETF 5652:2009, 3852:2004, 3369:2002, 2630:1999.

```
# openssl cms -encrypt -secretkeyid 07 \
              -secretkey 0123456789ABCDEF0123456789ABCDEF \
              -aes256 -in message.txt -out encrypted2.txt

# openssl cms -decrypt -in encrypted2.txt -secretkeyid 07 \
              -secretkey 0123456789ABCDEF0123456789ABCDEF
```

CMS Codec KEKRI: KEK+AES-KW+CBC:

```
def kekri(kekri, privateKeyBin, encOID, data, iv) do
    {:'KEKRecipientInfo',_vsn,_,{_,kea,_},encryptedKey} = kekri
    _ = CA.ALG.lookup(kea)
    {enc,_} = CA.ALG.lookup(encOID)
    unwrap = CA.AES.KW.unwrap(encryptedKey,privateKeyBin)
    res = CA.AES.decrypt(enc, data, unwrap, iv)
    {:ok, res}
end
def testDecryptKEK(), do: CA.CMS.decrypt(testKEK(), testPrivateKeyKEK())

def testPrivateKeyKEK() do
    {:kek, :binary.decode_hex("0123456789ABCDEF0123456789ABCDEF")}
end

def testKEK() do
    {:ok,base} = :file.read_file "priv/certs/encrypted2.txt"
    [_,s] = :string.split base, "\n\n"
    x = :base64.decode s
    :'CryptographicMessageSyntax-2010'.decode(:ContentInfo, x)
end
```

10.2.4 CMS-KTRI-RSA

The very first CMS IETF 3852:1999:

```
# gpgsm --list-keys
# gpgsm --list-secret-keys
# gpgsm -r 0xD3C8F78A -e CNAME > cms.bin
# gpgsm -u 0xD3C8F78A -d cms.bin
# gpgsm --export-secret-key-p12 0xD3C8F78A > key.bin
# openssl pkcs12 -in key.bin -nokeys -out public.pem
# openssl pkcs12 -in key.bin -nocerts -nodes -out private.pem
```

CMS Codec KTRI: RSA+RSAES-OAEP:

```
def ktri(ktri, privateKeyBin, encOID, data, iv) do
    {:'KeyTransRecipientInfo',_vsn,_,{_,schemeOID,_},key} = ktri
    {:rsaEncryption,_} = CA.ALG.lookup schemeOID
    {enc,_} = CA.ALG.lookup(encOID)
    sessionKey = :public_key.decrypt_private(key, privateKeyBin)
    res = CA.AES.decrypt(enc, data, sessionKey, iv)
    {:ok, res}
end
def testDecryptRSA(), do: CA.CMS.decrypt(testRSA(), testPrivateKeyRSA())

def testPrivateKeyRSA() do
    {:ok,bin} = :file.read_file("priv/rsa-cms.key")
    pki = :public_key.pem_decode(bin)
    [{:PrivateKeyInfo,_,_}] = pki
    rsa = :public_key.pem_entry_decode(hd(pki))
    {:'RSAPrivateKey',:'two-prime',_n,_e,_d,_,_,_,_,_,_} = rsa
    {:rsaEncryption,rsa}
end

def testRSA() do
    {:ok,x} = :file.read_file "priv/rsa-cms.bin"
    :'CryptographicMessageSyntax-2010'.decode(:ContentInfo, x)
end
```

10.2.5 KDF

KDF (MD5: 128, SHA: 160—512) and HKDF (HMAC) Key Derive functions used in ECC CMS schemes as of NIST SP 800-108r1.

```
defmodule KDF do
    def hl(:md5),    do: 16
    def hl(:sha),    do: 20
    def hl(:sha224), do: 28
    def hl(:sha256), do: 32
    def hl(:sha384), do: 48
    def hl(:sha512), do: 64

    def derive(h, d, len, x) do
        :binary.part(:lists.foldr(fn i, a ->
            :crypto.hash(h, d <> <> <> x) <> a
        end, <<>>, :lists.seq(1,round(Float.ceil(len/hl(h))))), 0, len)
    end
end
```

10.2.6 AES-KW

AES Key Wrap function is applicable to keys of 128/192/256 bit using AES-ECB encoding as of RFC 5649:2009 Advanced Encryption Standard (AES) Key Wrap with Padding Algorithm.

```
-define(MSB64,      1/unsigned-big-integer-unit:64).
-define(DEFAULT_IV, << 16#A6A6A6A6A6A6A6A6:?MSB64 >>).

unwrap(CipherText, KEK) -> unwrap(CipherText, KEK, ?DEFAULT_IV).
unwrap(CipherText, KEK, IV)
        when (byte_size(CipherText) rem 8) =:= 0
        andalso (bit_size(KEK) =:= 128
            orelse bit_size(KEK) =:= 192
            orelse bit_size(KEK) =:= 256) ->
    BlockCount = (byte_size(CipherText) div 8) - 1,
    IVSize = byte_size(IV),
    case do_unwrap(CipherText, 5, BlockCount, KEK) of
        << IV:IVSize/binary, PlainText/binary >> ->
            PlainText;
        _ ->
            erlang:error({badarg, [CipherText, KEK, IV]})
    end.

codec(128) -> aes_128_ecb;
codec(192) -> aes_192_ecb;
codec(256) -> aes_256_ecb.

do_unwrap(Buffer, J, _BlockCount, _KEK) when J < 0 -> Buffer;
do_unwrap(Buffer, J, BlockCount, KEK) ->
    do_unwrap(do_unwrap(Buffer, J, BlockCount, BlockCount, KEK),
              J - 1, BlockCount, KEK).
do_unwrap(Buffer, _J, I, _BlockCount, _KEK) when I < 1 -> Buffer;
do_unwrap(<< A0:?MSB64, Rest/binary >>, J, I, BlockCount, KEK) ->
    HeadSize = (I - 1) * 8,
    << Head:HeadSize/binary, B0:8/binary, Tail/binary >> = Rest,
    Round = (BlockCount * J) + I,
    A1 = A0 bxor Round,
    Data = << A1:?MSB64, B0/binary >>,
    << A2:8/binary, B1/binary >>
        = crypto:crypto_one_time(codec(bit_size(KEK)),
          KEK, ?DEFAULT_IV, Data, [{encrypt,false}]),
    do_unwrap(<< A2/binary, Head/binary, B1/binary,
          Tail/binary >>, J, I - 1, BlockCount, KEK).
```

10.2.7 AES-256

All AES-256 flavours are implemented for a wide range of ECC Key Agreement schemes.

```
def decrypt(crypto_codec, data, key, iv \\ :crypto.strong_rand_bytes(16))
def decrypt(:'id-aes256-ECB',data,key,iv), do: decryptAES256ECB(data,key,iv)
def decrypt(:'id-aes256-CBC',data,key,iv), do: decryptAES256CBC(data,key,iv)
def decrypt(:'id-aes256-GCM',data,key,iv), do: decryptAES256GCM(data,key,iv)
def decrypt(:'id-aes256-CCM',data,key,iv), do: decryptAES256CCM(data,key,iv)
def test() do
  [
    check_SECP384R1_GCM256(),
    check_X25519_GCM256(),
    check_C2PNB368w1_GCM256(),
    check_BrainPoolP512t1_GCM256(),
    check_BrainPoolP512t1_GCM256(),
    check_SECT571_GCM256(),
    check_X448_GCM256(),
    check_X448_CBC256(),
    check_X448_ECB256(),
  ]
end
```

10.3 Імплементація CMP сервера у складі АЦСК

IETF follow up (PKIX): 7030, 6960, 6818, 6844, 6712, 6664, 6402, 6277, 6170, 6024, 6025, 5934, 5912–5914, 5877, 5816, 5755, 5756, 5758, 5697, 5636, 5480, 5272–5274, 5280, 5055, 5019, 4985, 4683, 4630, 4476, 4387, 4325, 4158, 4210, 4211, 4055, 4043, 3874, 3779, 3820, 3739, 3709, 3628, 3161, 3029, 2797, 2559, 2587, 3039, 3029, 2511, 2510.

Compatibility: OpenSSL, Cisco, Red Hat, Siemens, Nokia, IBM.

Ця стаття могла би називати «Як написати CMP сервер за 30 хвилин», але на відміну від попередньої статті про LDAP, ця вже покриває більше ніж тузінь ASN.1 файлів, добре шо ми вже познайомилися з CMS та LDAP бібіліотеками та їх ASN.1 файлами. В цій статті про CMP нас в основному цікавитимуть PKIXCMP-2009, PKIXCRMF-2009 та EnrollmentMessageSyntax-2009 для CMC.

```
CMS-AES-CCM-and-AES-GCM-2009.asn1
CMSAesRsaesOaep-2009.asn1
CMSECCAlgs-2009-02.asn1
CMSECDHAlgs-2017.asn1
CryptographicMessageSyntax-2009.asn1
CryptographicMessageSyntax-2010.asn1
CryptographicMessageSyntaxAlgorithms-2009.asn1
EnrollmentMessageSyntax-2009.asn1
PKCS-10.asn1
PKCS-7.asn1
PKIX1Explicit-2009.asn1
PKIX1Implicit-2009.asn1
PKIXAlgs-2009.asn1
PKIXCMP-2009.asn1
PKIXCRMF-2009.asn1
```

10.3.1 CSR

Отже починається написання CMP серверу з найголовнішої його функції: видачі сертифікату по PCKS-10 CSR реквесту. Схема наступна: Клієнт генерує приватний ключ, конвертує його в PEM файл, відсилає як P10CR повідомлення у складі payload PKIMessage, отримує відповідь CP, після чого клієнт шле ще одне повідомлення CERTCONF, після якого CMP сервер повинен відповисти PKICONF повідомленням.

```
def csr(user) do
    {ca_key, ca} = read_ca()
    priv = X509.PrivateKey.new_ec(:secp384r1)
    der = :public_key.der_encode(:ECPrivateKey, priv)
    pem = :public_key.pem_encode([{:ECPrivateKey, der, :not_encrypted}])
    :file.write_file(user <> ".key", pem)
    :io.format '~p~n', [priv]
    csr = X509.CSR.new(priv, "/C=UA/L=Kyiv/O=SYNRC/CN=" <> user,
       extension_request: [
          X509.Certificate.Extension.subject_alt_name(["n2o.dev"])])
    :io.format 'CSR: ~p~n', [csr]
    :file.write_file(user <> ".csr", X509.CSR.to_pem(csr))
    true = X509.CSR.valid?(csr)
    subject = X509.CSR.subject(csr)
    :io.format 'Subject ~p~n', [subject]
    :io.format 'CSR ~p~n', [csr]
    X509.Certificate.new(X509.CSR.public_key(csr), subject, ca, ca_key,
       extensions: [subject_alt_name:
         X509.Certificate.Extension.subject_alt_name(["n2o.dev", "erp.uno"])
    ])
    csr
end
```

Перед початком роботи CMP сервера повинен бути згенерований рутовий CA сертифікат з приватним ключем, ці два файла ми зберігаємо на диск, і у всіх подальших операціях користуємося ними. Для генерації файлів використовуємо функцію CA.CSR.ca.

```
def ca() do
     ca_key = X509.PrivateKey.new_ec(:secp384r1)
     ca = X509.Certificate.self_signed(ca_key,
           "/C=UA/L=Kyiv/O=SYNRC/CN=CSR-CMP", template: :root_ca)
     der = :public_key.der_encode(:ECPrivateKey, ca_key)
     pem = :public_key.pem_encode([{:ECPrivateKey, der, :not_encrypted}])
     :file.write_file "ca.key", pem
     :file.write_file "ca.pem", X509.Certificate.to_pem(ca)
     {ca_key, ca}
 end

 def read_ca() do
     {:ok, ca_key_bin} = :file.read_file "ca.key"
     {:ok, ca_bin} = :file.read_file "ca.pem"
     {:ok, ca_key} = X509.PrivateKey.from_pem ca_key_bin
     {:ok, ca} = X509.Certificate.from_pem ca_bin
     {ca_key, ca}
 end
```

Для одноразовоїї генерації серверних сертифікатів які обсуговують клієнтські TLS сесії можна використати наступний код.

```
 def server(name) do
     {ca_key, ca} = read_ca()
     server_key = X509.PrivateKey.new_ec(:secp384r1)
       X509.Certificate.new(X509.PublicKey.derive(server_key),
          "/C=UA/L=Kyiv/O=SYNRC/CN=" <> name, ca, ca_key,
          extensions: [subject_alt_name:
             X509.Certificate.Extension.subject_alt_name(["n2o.dev", "erp.uno"
     ]) ])
 end
```

10.3.2 CMS

Детально сімейство протоколів і CMS кодування описано в окремій статті присвяченій CMS Compliance. CMS кодування використовується тільки для CMC сервера, тому ми це поки висвітлювати не будемо.

10.3.2.1 CMP/CSR/TCP

RFC 6712, 4210. Для початку напишемо простий PKIMessage сервер.

```
defmodule CA.CMP do
  @moduledoc "CA/CMP TCP server."
  require CA

  def start(), do: :erlang.spawn(fn -> listen(1829) end)
  def listen(port) do
      {:ok, socket} = :gen_tcp.listen(port,
        [:binary, {:packet, 0}, {:active, false}, {:reuseaddr, true}])
      accept(socket)
  end

  def accept(socket) do
      {:ok, fd} = :gen_tcp.accept(socket)
      :erlang.spawn(fn -> __MODULE__.loop(fd) end)
      accept(socket)
  end

  def loop(socket) do
      case :gen_tcp.recv(socket, 0) do
           {:ok, data} ->
               [headers,body] = :string.split data, "\r\n\r\n", :all
               {:ok,dec} = :'PKIXCMP-2009'.decode(:'PKIMessage', body)
               {:PKIMessage, header, body, code, _extra} = dec
               __MODULE__.message(socket, header, body, code)
               loop(socket)
          {:error, :closed} -> :exit
      end
  end
```

10.3.2.2 PKIMessage.protection

Розберемося з полем PKIMessage.protection, в якому зберігається результат PBKDF2 алгоритма. Майте на увазі шо OpenSSL за замовчування використовує 20-байтні ключі та HMAC/SHA-1 у якості MAC функції, хоча OWF в 500 ітераціях обчислюється за допомогою OWF функції SHA-256.

10.3.2.3 ANSWER

Оскільки CMP сервер повинен працювати по HTTP/1.0 згідно стандартів додаємо необхідні HTTP заголовки.

```
def answer(socket, header, body, code) do
     message = CA."PKIMessage"(header: header, body: body, protection: code)
     {:ok, bytes} = :'PKIXCMP-2009'.encode(:'PKIMessage', message)
     res =  "HTTP/1.0 200 OK\r\n"
         <> "Server: SYNRC CA/CMP\r\n"
         <> "Content-Type: application/pkixcmp\r\n\r\n"
         <> :erlang.iolist_to_binary(bytes)
     :gen_tcp.send(socket, res)
 end
```

10.3.2.4 P10CR/CP

Запускаємо сервер та генеруємо сертифікати CA та CSR користувача:

```
. iex -S mix
> CA.CSR.ca
> CA.CSR.csr "maxim"
```

Запускаємо клієнтський запит за допомогою OpenSSL:

```
# openssl cmp -cmd p10cr -server localhost:1829 \
#             -path . -srvcert ca.pem -ref cmptestp10cr \
#             -secret pass:0000 -certout . client.csr
```

Пишемо функцію видачі сертифікату:

```
def message(socket, header, {:p10cr, csr} = body, code) do
      {:PKIHeader, pvno, from, to, messageTime, protectionAlg,
         _senderKID, _recipKID, transactionID, senderNonce,
         _recipNonce, _freeText, _generalInfo} = header
      true = code == validateProtection(header, body, code)

      {ca_key, ca} = CA.CSR.read_ca()
      subject = X509.CSR.subject(csr)
      :io.format '~p~n',[subject]
      true = X509.CSR.valid?(CA.parseSubj(csr))
      cert = X509.Certificate.new(X509.CSR.public_key(csr),
         CA.CAdES.subj(subject), ca, ca_key,
         extensions: [subject_alt_name:
            X509.Certificate.Extension.subject_alt_name(["synrc.com"]) ])

      reply = CA."CertRepMessage"(response:
            [ CA."CertResponse"(certReqId: 0,
              certifiedKeyPair: CA."CertifiedKeyPair"(certOrEncCert:
                {:certificate, {:x509v3PKCert, CA.convertOTPtoPKIX(cert)}}),
              status: CA."PKIStatusInfo"(status: 0))])

      pkibody = {:cp, reply}
      pkiheader = CA."PKIHeader"(sender: to, recipient: from, pvno: pvno,
          recipNonce: senderNonce, transactionID: transactionID,
          protectionAlg: protectionAlg, messageTime: messageTime)
      answer(socket, pkiheader, pkibody,
          validateProtection(pkiheader, pkibody, code))
  end
```

10.3.2.5 CERTCONF/PKICONF

```
def message(socket, header, {:certConf, statuses}, code) do
      {:PKIHeader, _, from, to, _, _, _, _, _, senderNonce, _, _, _} = header

      :lists.map(fn {:CertStatus,bin,no,{:PKIStatusInfo, :accepted, _, _}} ->
          :logger.info 'CERTCONF ~p request ~p~n', [no,:binary.part(bin,0,8)]
      end, statuses)

      pkibody = {:pkiconf, :asn1_NOVALUE}
      pkiheader = CA."PKIHeader"(header, sender: to, recipient: from,
          recipNonce: senderNonce)
      answer(socket, pkiheader, pkibody,
          validateProtection(pkiheader, pkibody, code))
  end
```

В результаті в консолі повинні спостерігати:

```
CMP info: sending P10CR
CMP info: received CP
CMP info: sending CERTCONF
CMP info: received PKICONF
CMP info: received 1 enrolled certificate(s), saving to file 'maxim.pem'
```

10.3.2.6 GENM/GENP

Далі можете написати інші функції:

```
# openssl cmp -cmd genm -server 127.0.0.1:1829 \
#             -recipient "/CN=CMPserver" -ref 1234 -secret pass:0000


def message(_socket, _header, {:genm, req} = _body, _code) do
      :io.format 'generalMessage: ~p~n', [req]
  end
```

10.3.2.7 IR/IP

```
# openssl cmp -cmd ir -server 127.0.0.1:1829 \
#             -path . -srvcert ca.pem -ref NewUser \
#             -secret pass:0000 -certout maxim.pem \
#             -newkey maxim.key -subject "/CN=maxim/O=SYNRC/ST=Kyiv/C=UA"
```

```
 def message(_socket, _header, {:ir, req}, _) do
      :lists.map(fn {:CertReqMsg, req, sig, code} ->
         :io.format 'request: ~p~n', [req]
         :io.format 'signature: ~p~n', [sig]
         :io.format 'code: ~p~n', [code]
      end, req)
  end
```

10.3.2.8 CR/CP

```
# openssl cmp -cmd cr -server 127.0.0.1:1829 \
#             -path . -srvcert ca.pem -ref NewUser \
#             -secret pass:0000 -certout maxim.pem \
#             -newkey maxim.key -subject "/CN=maxim/O=SYNRC/ST=Kyiv/C=UA"
```

10.3.2.9 Висновки

```
defmodule CA do
  use Application
  use Supervisor

  require Record
  Enum.each(Record.extract_all(from_lib: "ca/include/PKIXCMP-2009.hrl"),
            fn {name, definition} -> Record.defrecord(name, definition) end)

  Enum.each(Record.extract_all(from_lib: "public_key/include/public_key.hrl"),
            fn {name, definition} -> Record.defrecord(name, definition) end)

  def init([]), do: {:ok, { {:one_for_one, 5, 10}, []} }
  def start(_type, _args) do
      :logger.add_handlers(:ldap)
      CA.CMP.start
      CA.CMC.start
      :supervisor.start_link({:local, __MODULE__}, __MODULE__, [])
  end
```

Ну PasswordBasedMac в нас є, тепер треба DHMac, але shared secret можна і в PBM засунути. Є ше Proof Of Posession (POP) — там зразу ECDSA verify. Я до речі думаю в CA тримати ключі для всіх кривих, і коли я виставлятиму сервіс то я буду виставляти його на N портах і N ключах, щоб будь який клієнтський TLS сертифікат приймався як рідний! Chat X.509 дає можливість вибирати TLS сертифікати автоматично по обраних кривих. Ви вибираєте під якими ключами сьогодні заходити. LDAP, MQTT, NS, CA — в кожного сервісу свої N портів і N серверних TLS сертифікатів. Передбачається що перший сертифікат видається DH по TCP а потім зразу всьо переходить в TLS режим і всі наступні сертифікати вже видаються всередині клієнтського TLS. При реєстрації користувач зразу доступний в LDAP якшо захотів зробити себе відкритим для пошуку. Після реєстрації пошук в директорії і френдування (обмін ключами) і поїхали чат в обгортках CAdES, CMS, ECDSA/AES — лейби біля повідомлень ПІДПИС/ШИФР.

10.3.3 CA, АЦСК, ЦЗО та ОЗО

10.4 Безпечна система доменних імен DNSSEC

10.5 Система директорії підприємства LDAP

Пройшло 13 років з того часу як компанія SYNRC випустила була свій власний брендований LDAP Directory Server на Erlang з підтримкою MongoDB. Взагалі баз які підтримують префіксний і суфіксний пошук по B-Tree таблицям не так багато, в основному це складні SQL та MongoDB. Хоча такі бази як Mnesia, LMDB, BDB-похідні (RocksDB, LevelDB) немають повнотекстового пошуку, це можна реалізувати шляхом повного траверса, що на цих базах зазвичай працює швидко (якшо на C), і ше швидше якшо база підтримує mmap (LMDB).

Непереможний по перформенсу станом на зараз є OpenLDAP (LMDB), однак нам хочеться мати свою імплементацію, яку можна було би використовувати як фірмове сховище даних для директорії ресурсів підприємства згідно як міжнародних стандартів так і стандартів України. Ми захотіли відмовитися від зовнішньої бази даних (MongoDB) що ускладнює розробку, і хотілося вибрати шось вбудовуване для Erlang, вибирали між zambal/elmdb та elixir-sqlite/exqlite, переміг SQLite тому шо хотілося мати мінімальну ідіоматичну імплементацію, така шоб з одного боку була зрозуміла навіть людям без глибокої освіти в Computer Science, а з іншого боку відкривала двері в підтримку інших промислових корпоративних SQL джерел даних (Oracle, T-SQL, PostgreSQL).

На відміну від попередньої версії SYNRC LDAP яка робилася під стартап PEOPLE|SYNC який ми хотіли продати PEOPLE|NET, а потім Київстар, як SyncML стартап по синхронізації контактних книг, ця версія робиться для стартапа SYNRC CHAT для розвідки. В цій версії ми значну увагу приділили сумісністю з клієнтами, такими як Apache Directory Studio, а також усіма LDIF файлами які ми змогли знайти в інтернеті.

Так як якісну безпечну розподілену інфраструктуру яка відповідає міжнародним і українським стандартами неможливо створити без CA/CMS, OCSP, TSP, LDAP, DNS/DNSSEC, MQTT серверів, то всі ці сервери є фундаментом продуктів SYNRC які формують перший рівень фреймворку Сохацького який умовно називається Security або Безпека Підприємства. На цьому фреймворку побудовані головні інфраструктурні елементи країни, а також реєстрові системи.

Загалом, LDAP це дуже древня і надійна технологія яка присутня буквально у всіх топових компаніях, корпораціях, великих і малих бізнесах. Так, є сучасній Identity Server продукти (Hashicorp) які не підтримуються LDAP, але це поодинокі непопулярні маргінальні екземпляри. SYNRC LDAP це гарний початок для малих, середніх і великих бізнесів навести порядок в штатних розкладах, календарях, задачах, ресурсах підприємства, таких як автопарки, IoT, реєстрах персональних і проми-

слових комп'ютерів, портів, правил ABAC, правил маршрутизації, контактних книгах користувачів, одним словом все для чого створений і де використовується LDAP.

10.5.1 Вертикальні бази

Я вперше познайомився з вертикальними базами коли працював в International Land Systems, Inc. Тоді в нас була система документообігу на вертикальній базі, які дуже часто використовуються в системах документообігу. Наприклад таку схему даних використовує Alfresco, а також SQL розширення для зберігання XML у Oracle та інших SQL базах.

Основна ідея вертикальних баз, або схем, полягає в тому шо об'єкти зберігаються не у плоских таблицях де кожен атрибут це окрема колонка, а всі атрибути та їх значення зберігаються в трьох колонках, де перша — це номер об'єкта, друга — ім'я атрибута, а третя — значення атрибуту. Це дозволяє тримати розріжені (атрибутами) об'єкти, та певним чином спростити управління базою. Всі наївні імплементації вертикальних баз страждаються по перформенсу, тому потрібно бути обрежним. Наприклад, ми не рекомендуємо використовувати SYNRC LDAP де об'єм директоріє більше ніж пів мільйона співробітників, наприклад для Walmart. Для великих корпорацій краще брати OpenLDAP.

10.5.2 Предметна область

10.5.2.1 Netscape, Sun DS, 389 DS, Oracle

PEOPLE|SYNC стартап SYNRC 2007 року підтримував також роботу з Sun Directory Server. Його родовід бере початок від сервера OpenLDAP, в 1996 року стартував форк Netscape Directory Server. Після банкротства Netscape право на код викупила компанія AOL, яка ліцензувала право на розробку компанії Sun Microsystems, зберігши право на код. В 2009 році Sun DS був переіменований на 389 Directory Server, а Oracle почала свій форк Sun DS під назвою Oracle Directory Server. 389 Directory Server також можна зустріти під іменами Fedora DS та Red Hat DS, оскільки це основні донори проекту.

10.5.2.2 Microsoft Active Directory

Традиційно кожна корпорація займається розробкою свого LDAP сервера, компанія Microsoft випустила свій перший в 1999 році. Active Directory у якості бекенда використовує Extensible Storage Engine ESENT.DLL, також відомий як JetDB, на яцій побудований також Windows Registry, Microsoft Access, та можливо і інші внутрішні продукти компанії Microsoft.

10.5.2.3 OpenLDAP, Apple Open Directory

Були часи, що OpenLDAP був вбудований в кожну версію Mac OS X, але Apple почала розвивати свій власний Open Directory Server, оскільки безпека кожної корпорації полягає у тому числі в брендованих LDAP серверах під свої потреби та політику розробки..

10.5.3 TCP сервер

Я неодноразово використовував написання LDAP серверу у своїх Erlang курсах, а також на конференціях у якості майстер-класу по програмуванню, де ми з аудиторією пишемо всі разом LDAP сервер за 45 хвилин з моїми коментарями та інтеракцією. Рекорд на відео був поставлений 30 хвилин, так шо це не прікол, я дійсно MVP всіх продуктів SYNRC можу написати за 30 хвилин кожен. Власне це є одним з критеріїв SYNRC, що тісно переплетено з показником LOC. Ця версія SYNRC LDAP з підтримкою SQLite займає 300 рядків коду і проходить всі LDIF тест сюїти.

Перед початком поставте Erlang та його Erlang AST фронтенд Elixir. Ставити можна завжди тільки Elixir, Erlang піде як залежність в будь-якому пекедж менеджері.

```
# apt install elixir
```

Створюємо папку проекту і в ній створюємо файл mix.exs для уніфікованого депенденсі і пекадж менеджера Erlang і Elixir мов, mix.

Крім класичної прелюдії mix.exs, нам цікавий параметр exqlite, це ім'я бібліотеки пакетного менеджера hex.pm, яка містить в собі 8-мегабайтний Ci файл, і FFI обгортку для нього яка в Erlang світі називається NIF.

```
defmodule LDAP.Mixfile do
  use Mix.Project
  def project(), do:
    [ app: :ldap, version: "8.7.20", deps: deps(),
      releases: [ ldap: [ include_executables_for: [:unix],
                          cookie: "SYNRC:LDAP" ] ] ]
  def application(), do:
    [ mod: {LDAP, []},
      extra_applications: [ :eldap, :asn1 ] ] end
  def deps(), do:
    [ {:exqlite, "~> 0.13.14"} ]
end
```

Далі пишемо найпростіший ідіоматичний Erlang TCP сервер. Довгий час я використовував класичні, розширені версії на недокументованій функції prim ,5/,,,,.!

Якшо в config/config.exs немає параметра ldap:intance то створюється нова SQLite база по рендомному хешу з налаштуваннями по перформансу: 1) відключений журнал, 2) ін-меморі буфер, 3)

великий кеш, 4) примусова синхронність; які визначаються відповідними SQL прагмами.

Архітектура TCP сервера відповідає POSIX, ми створюємо лістенер, який на кожне вхідне TCP повідомлення стартує акцептори, які стартують некотрольований Erlang процес — лупер, який обслуговує вхідне TCP повідомлення. Для декодування використовується згенерований ASN.1 компілятором енкодер/декодер LDAP протоколу по файлу LDAP.ans1, який можна знайти прямо в RFC IETF на нормативно-правових актах України.

```
# erlc LADP.asn1
```

Після геренації покладіть файли в LDAP.erl та LDAP.hrl в папку src проекту. А в папці lib створіть обгортку для згенерованих рекордів за допомогою Record,

```
defmodule DS do
    require Record
    Enum.each(Record.extract_all(from_lib: "ldap/include/LDAP.hrl"),
            fn {name, definition} -> Record.defrecord(name, definition) end)
end
```

а також створіть козу Erlang аплікейшина в Elixir синтаксисі, файл ldap.ex:

```
defmodule LDAP do
   import Exqlite.Sqlite3
   require DS
   use Application
   use Supervisor
   def code(), do: :binary.encode_hex(:crypto.strong_rand_bytes(8))
   def init([]), do: {:ok, { {:one_for_one, 5, 10}, []} }
   def start(_, _) do
       :logger.add_handlers(:ldap)
       :supervisor.start_link({:local, LDAP}, LDAP, [])
   end
   def initDB(path) do
       {:ok, conn} = open(path)
       :logger.info 'SYNRC LDAP Instance: ~p', [path]
       :logger.info 'SYNRC LDAP Connection: ~p', [conn]
       execute(conn, "create table ldap (rdn text,att text,val binary)")
       :ok = execute(conn, "PRAGMA journal_mode = OFF;")
       :ok = execute(conn, "PRAGMA temp_store = MEMORY;")
       :ok = execute(conn, "PRAGMA cache_size = 1000000;")
       :ok = execute(conn, "PRAGMA synchronous = 0;")
       conn
   end
   def listen(port,path) do
       conn = initDB(path)
       {:ok, socket} = :gen_tcp.listen(port,
         [:binary, {:packet, 0}, {:active, false}, {:reuseaddr, true}])
       accept(socket,conn)
   end
   def accept(socket,conn) do
       {:ok, fd} = :gen_tcp.accept(socket)
       :erlang.spawn(fn -> loop(fd, conn) end)
       accept(socket,conn)
   end
```

```
    def start() do
        :erlang.spawn(fn ->
            listen(:application.get_env(:ldap,:port,1489),
                   :application.get_env(:ldap,:instance,code())) end)
    end
    def answer(response, no, op, socket) do
        message = DS."LDAPMessage"(messageID: no, protocolOp: {op, response})
        {:ok, bytes} = :'LDAP'.encode(:'LDAPMessage', message)
        send = :gen_tcp.send(socket, :erlang.iolist_to_binary(bytes))
    end
    def loop(socket, db) do
        case :gen_tcp.recv(socket, 0) do
             {:ok, data} ->
                  case :'LDAP'.decode(:'LDAPMessage',data) do
                       {:ok,decoded} ->
                           {:'LDAPMessage', no, payload, _} = decoded
                           message(no, socket, payload, db)
                           loop(socket, db)
                       {:error,reason} ->
                          :logger.error 'ERROR: ~p', [reason]
                         :exit
                  end
             {:error, :closed} -> :exit
        end
    end
end
```

Цей сервер вже може відповідати на запити ldapmodify але буде їх блокувати так як поки що не відповідає належним чином. Запустити програму можна класичними мантрами Elixir, а в Elixir Shell виконати функцію запуску TCP лістенера, для цього перевпевніться що дефаултний порт 1389 вільний.

```
# mix deps.get
# iex -S mix
> LDAP.start
#PID<0.311.0>
iex(2)>
04:58:26.030 [info]  SYNRC LDAP Instance: "416C4C41ED2C7060"
04:58:26.030 [info]  SYNRC LDAP Connection: #Reference<
     0.1146704550.396492828.212314>
iex(3)>
nil
```

10.5.3.1 BIND

Для удачного демо я раджу починати з функції BIND. Для цього створимо в базі записи по яким будемо аутентифікуватися.

```
createDN(conn, "dc=synrc,dc=com",
    [ attr("dc",["synrc"]), attr("objectClass",["top","domain"]) ])
createDN(conn, "ou=schema",
    [ attr("ou",["schema"]), attr("objectClass",["top","domain"]) ])
createDN(conn, "cn=tonpa,dc=synrc,dc=com",
    [ attr("cn",["tonpa"]),attr("uid",["1000"]),
      attr("objectClass",["inetOrgPerson","posixAccount"]) ])
createDN(conn, "cn=rocco,dc=synrc,dc=com",
    [ attr("cn",["rocco"]),attr("uid",["1001"]),
      attr("objectClass",["inetOrgPerson","posixAccount"]) ])
createDN(conn, "cn=admin,dc=synrc,dc=com",
    [ attr("rootpw",["secret"]), attr("cn",["admin"]),
      attr("objectClass",["inetOrgPerson"]) ])

def appendNotEmpty([]),  do: []
def appendNotEmpty(res) do
    res ++ case res do [] -> [] ; _ -> ',' end
end
def createDN(db, dn, attributes) do
    norm  = :lists.foldr(fn {:PartialAttribute, att, vals}, acc ->
            :lists.map(fn val -> [qdn(dn),att,val] end, vals) ++
                                 acc end, [], attributes)
    {_,p} = :lists.foldr(fn x, {acc,res} ->
             {acc + length(x), appendNotEmpty(res)
              ++ :io_lib.format('(?~p,?~p,?~p)', [acc+1,acc+2,acc+3])}
            end, {0,[]}, norm)
    {:ok, statement} = prepare(db,
          'insert into ldap (rdn,att,val) values ' ++ p ++ '')
    :ok = bind(db, statement, :lists.flatten(norm))
    :done = step(db, statement)
end
def message(no, socket, {:bindRequest, {_,_,bindDN,{:simple, password}}}, db)
   do
    sql = "select rdn, att from ldap where " <>
          "rdn = ?1 and att = 'rootpw' and val = ?2"
    {:ok, statement} = prepare db, sql
    bind(db, statement, [hash(qdn(bindDN)),password])
    case step(db, statement) do
        :done ->  code = :invalidCredentials
                  :logger.error 'BIND Error: ~p', [code]
                  response = DS."BindResponse"(resultCode: code,
                      matchedDN: "", diagnosticMessage: 'ERROR')
                  answer(response, no, :bindResponse, socket)
         {:row,[dn,password]} ->
                  :logger.info 'BIND DN: ~p', [bindDN]
                  response = DS."BindResponse"(resultCode: :success,
                       matchedDN: "", diagnosticMessage: 'OK')
                  answer(response, no, :bindResponse, socket)
    end
end
def message(no, socket, {:bindRequest, {_,_,bindDN,creds}}, db) do
    code = :authMethodNotSupported
    :logger.info 'BIND ERROR: ~p', [code]
    response = DS."BindResponse"(resultCode: code,
       matchedDN: "", diagnosticMessage: 'ERROR')
    answer(response, no, :bindResponse, socket)
end
```

10.5.3.2 ADD

```
def message(no, socket, {:addRequest, {_,dn, attributes}}, db) do
      {:ok, statement} = prepare(db, "select rdn, att, val from ldap where rdn =
       ?1")
      bind(db, statement, [hash(qdn(dn))])
      case step(db, statement) do
            {:row, _} ->
                :logger.info 'ADD ERROR: ~p', [dn]
                resp = DS.'LDAPResult'(resultCode: :entryAlreadyExists,
                       matchedDN: dn, diagnosticMessage: 'ERROR')
                answer(resp, no, :addResponse, socket)
            :done ->
                createDN(db, dn, attributes)
                :logger.info 'ADD DN: ~p', [dn]
                resp = DS.'LDAPResult'(resultCode: :success,
                       matchedDN: dn, diagnosticMessage: 'OK')
                answer(resp, no, :addResponse, socket)
       end
   end
```

10.5.3.3 DSE

Якшо тестувати наш прото-сервер не ldapmodify, а Apache Directory Studio, то вона початково буде питати так званий Root DSE об'єкт запитуючи після bind, search реквест з пустим DN. Для коректної репрезентації спеціалізованої інформації ми прошиємо стандартну відповідь на цей запит для нашого фірмового SYNRC LDAP сервера версії 2.0 який підтримує протокол LDAPv3. Він підтримує SIMPLE спосіб аунтентифікації тільки (поки що) і два іменних простори ключів: dc=synrc,dc=com ou=schema, як вимагає декілька RFC. Це всьо пакується у LDAPMessage і відправляється на клієнт функцією answer.

```
def attr(k,v),      do: {:PartialAttribute, k, v}
   def node(dn,attrs), do: {:SearchResultEntry, dn, attrs}

   def message(no, socket,
       {:searchRequest, {_,"",scope,_,limit,_,_,filter,attributes}}, db) do

       :logger.info 'DSE Scope: ~p', [scope]
       :logger.info 'DSE Filter: ~p', [filter]
       :logger.info 'DSE Attr: ~p', [attributes]

       :lists.map(fn response -> answer(response,no,:searchResEntry,socket) end,
         [ node("", [
              attr("supportedLDAPVersion", ['3']),
              attr("namingContexts", ['dc=synrc,dc=com','ou=schema']),
              attr("supportedControl", ['1.3.6.1.4.1.4203.1.10.1']),
              attr("supportedExtensions", ['1.3.6.1.4.1.4203.1.11.3']),
              attr("altServer", ['ldap.synrc.com']),
              attr("subschemaSubentry", ['ou=schema']),
              attr("vendorName", ['SYNRC LDAP']),
              attr("vendorVersion", ['2.0']),
              attr("supportedSASLMechanisms", ['SIMPLE']),
              attr("objectClass", ['top','extensibleObject']),
              attr("entryUUID", [code()])]),
              attr("supportedFeatures", [ '1.3.6.1.1.14',
                                          '1.3.6.1.4.1.4203.1.5.1'])
         ])

       resp = DS.'LDAPResult'(resultCode: :success, matchedDN: "",
      diagnosticMessage: 'OK')
       answer(resp, no, :searchResDone,socket)
   end
```

Після цього зможуть розгортати в інтерфейсі дерево об'єктів.

10.5.3.4 MODIFY

```
def modifyDN(db, dn, attributes), do:
    :lists.map(fn {_, :add, x}     -> modifyAdd(db,dn,x)
                  {_, :replace, x} -> modifyReplace(db,dn,x)
                  {_, :delete, x}  -> modifyDelete(db,dn,x) end, attributes)

def modifyAdd(db, dn, {_,att,[val]}) do
    {:ok, st} = prepare(db, "insert into ldap (rdn,att,val) values (?1,?2,?3)
   ")
    :logger.info 'MOD ADD RDN: ~p', [hash(qdn(dn))]
    bind(db, st, [hash(qdn(dn)),att,val])
    step(db,st)
end

def modifyReplace(db, dn, {_,att,[val]}) do
    {:ok, st} = prepare(db, "update ldap set val = ?1 where rdn = ?2 and att
   = ?3")
    :logger.info 'MOD REPLACE RDN: ~p', [hash(qdn(dn))]
    bind(db, st, [val,hash(qdn(dn)),att])
    step(db,st)
end

def modifyDelete(db, dn, {_,att,_}) do
    {:ok, st} = prepare(db, "delete from ldap where rdn = ?1 and att = ?2")
    :logger.info 'MOD DEL RDN: ~p', [hash(qdn(dn))]
    bind(db, st, [hash(qdn(dn)),att])
    res = step(db,st)
    collect0(db,st,res,[])
end

def message(no, socket, {:modifyRequest, {_,dn, attributes}}, db) do
   {:ok, statement} = prepare(db, "select rdn, att, val from ldap where rdn =
    ?1")
   bind(db, statement, [hash(qdn(dn))])
   case step(db, statement) do
        {:row, _} -> :logger.info 'MOD DN: ~p', [dn]
                     modifyDN(db, dn, attributes)
                     resp = DS.'LDAPResult'(resultCode: :success,
                         matchedDN: dn, diagnosticMessage: 'OK')
                     answer(resp, no, :modifyResponse, socket)
        :done ->     :logger.info 'MOD ERROR: ~p', [dn]
                     resp = DS.'LDAPResult'(resultCode: :noSuchObject,
                        matchedDN: dn, diagnosticMessage: 'ERROR')
                     answer(resp, no, :modifyResponse, socket)
   end
end
```

10.5.3.5 MODIFY DN

```
   def modifyRDN(socket, no, db, dn, new, del) do
       {:ok, st} = prepare(db, "update ldap set rdn = ?1 where rdn = ?2")
       :logger.info 'MODIFY RDN UPDATE: ~p', [hash(qdn(dn))]
       bind(db, st, [new,hash(qdn(dn))])
       step(db,st)
   end

   def message(no, socket, {:modDNRequest, {_,dn,new,del,_}}, db) do
       :logger.info 'MOD RDN DN: ~p', [dn]
       :logger.info 'MOD RDN newRDN: ~p', [new]
       :logger.info 'MOD RDN deleteOldRDN: ~p', [del]
       modifyRDN(socket, no, db, dn, new, del)
       resp = DS.'LDAPResult'(resultCode: :success,
           matchedDN: dn, diagnosticMessage: 'OK')
       answer(resp, no, :modDNResponse, socket)
   end
```

10.5.3.6 DELETE

```
def deleteDN(db, dn) do
       {:ok, st} = prepare(db, "delete from ldap where rdn = ?1")
       bind(db, st, [hash(qdn(dn))])
       res = step(db,st)
       collect0(db,st,res,[])
   end

   def message(no, socket, {:delRequest, dn}, db) do
       :logger.info 'DEL DN: ~p', [dn]
       deleteDN(db, dn)
       resp = DS.'LDAPResult'(resultCode: :success, matchedDN: dn,
      diagnosticMessage: 'OK')
       answer(resp, no, :delResponse, socket)
   end
```

10.5.3.7 SEARCH

10.5.3.8 COMPARE

```
def message(no, socket, {:compareRequest, {_,dn, assertion}}, db) do
       :logger.info 'CMP DN: ~p', [dn]
       :logger.info 'CMP Assertion: ~p', [assertion]
       result = compareDN(db, db, assertion)
       resp = DS.'LDAPResult'(resultCode: :success, matchedDN: dn,
     diagnosticMessage: 'OK')
       answer(resp, no, :compareResponse, socket)
   end
```

10.5.3.9 ABANDON/UNBIND

```
   def message(no, socket, {:abandonRequest, _}, db), do: :gen_tcp.close(socket)
   def message(no, socket, {:unbindRequest,  _}, db), do: :gen_tcp.close(socket)
```

10.5.4 Висновки

У цій статті ми переконалися, що можливо написати LDAP сервер на 300 рядків, а також що SQLite підходить для малих підприємств у якості сховища даних. Ми взяли повний контроль над продуктом, який не містить залежностей, що функціонують за межами контексту віртуальної машини, а також спростили процес розробки більш складних систем на базі цього продукту. Продукт буде корисний для апробації в підприємства зі стандартизованими та уніфікованими політиками управління ресурсами підприємствах, в телекомунікаційних продуктах, комунікаторах, месенджерах, тощо.

10.6 ASN.1 Компілятор

Питання вибору серіалізатора залежить від типу інфраструктури в компанії. Якшо ми говоримо про гомогенні інтерфейси та бібліотеки прикладного програмування то зазвичай вони достатньо розвинені аби автоматично генерувати структури для основних мов програмування для яких анонсується серіалізатор. Хоча формально йдеться про AST трансформацію з мови визначення типів структур в цільову мову програмування, такі системи називаються компіляторами мови визначення даних.

У світі існує багато рантаймів, і кожен з них пропонує свій рідний серіалізатор, який в незмінному вигляді представляє дані які циркулюють в рантаймі або віртуальній машині. Так на мовах .NET, Java, Haskell, Erlang є такі природні серіалазотори. В гомогенних архітектурах, де все побудовано навколо однії мови програмування, прийнято використовувати цей серіалізатор як основний для всіх сервісів написаних навіть на різних мовах програмування, так як BERT-RPC протокол який використовувався в Github.

Окремо виділяються формати які не прив'язані до однієї мови, а пропонуються як універсальні серіалізатори, такі як IDL COM/DCOM, ASN.1 DER, Erlang BERT/ETF, GRPC Gproto, SOAP XML/WBXML, та інші бінарні мови і серіалізатори.

Всі сертифікати в браузері, SSH ключі, PGP/GPG ключі, закриті і відкриті конверти, криптографічні повідомлення CMS, протоколи видачі сертифікатів, LDAP директорія та ще багато іншого включачи GSM/LTE та майже всі телекомунікаційні повідомлення визначаються за допомогою ASN.1.

10.6.1 Компілятори

Я хотів зробити огляд ASN.1 компіляторів, але з представлених безкоштовних жоден крім Erlang-ового не компілює повний набір ASN.1 файлів проекта SYNRC CA. Ні C-шний кацапський який використовується в Apple ASN1C, Ні F-повий ASN1SCC. Залишається сподіватися, що генератор C-шного коду Фабріса Белара ASN1CC зможе це зробити однак з огляду на його клієнтів навряд чи ми про це дізнаємося, так шоб прямо розказати в блозі.

Я подивився на помилки представлених на сайті ITU компіляторів і складається враження що сумісність з самим ж файлами дефініціями ITU ніхто не перевіряє навіть мануально. Просто вивішують в залікову таблицю будь-кого хто успішно розпарсав сабсет ASN.1 і згенерував якийсь граничні імплементація для свого випадку.

Тестовий набір файлів

Ось текстовий сет з сайту ITU яким я перевіряв компілятори.

```
AESKeyWrapWithPad-02.asn1
AESKeyWrapWithPad-88.asn1
ANSI-X9-42.asn1
ANSI-X9-62.asn1
AlgorithmInformation-2009.asn1
AttributeCertificateVersion1-2009.asn1
AuthenticationFramework.asn1
BasicAccessControl.asn1
CHAT.asn1
CMS-AES-CCM-and-AES-GCM-2009.asn1
CMSAesRsaesOaep-2009.asn1
CMSECCAlgs-2009-02.asn1
CMSECDHAlgs-2017.asn1
CertificateExtensions.asn1
Character-Coding-Attributes.asn1
Character-Presentation-Attributes.asn1
Character-Profile-Attributes.asn1
Colour-Attributes.asn1
CryptographicMessageSyntax-2009.asn1
CryptographicMessageSyntax-2010.asn1
CryptographicMessageSyntaxAlgorithms-2009.asn1
DOR-definition.asn1
Default-Value-Lists.asn1
DirectoryAbstractService.asn1
Document-Profile-Descriptor.asn1
EnrollmentMessageSyntax-2009.asn1
ExtendedSecurityServices-2009.asn1
External-References.asn1
Geo-Gr-Coding-Attributes.asn1
Geo-Gr-Presentation-Attributes.asn1
Geo-Gr-Profile-Attributes.asn1
ISO-STANDARD-9541-FONT-ATTRIBUTE-SET.asn1
ISO9541-SN.asn1
Identifiers-and-Expressions.asn1
InformationFramework.asn1
KEP.asn1
LDAP.asn1
Layout-Descriptors.asn1
Link-Descriptors.asn1
Location-Expressions.asn1
Logical-Descriptors.asn1
MultipleSignatures-2010.asn1
OCSP.asn1
PKCS-10.asn1
PKCS-12.asn1
PKCS-5.asn1
PKCS-7.asn1
PKCS-8.asn1
PKCS-9.asn1
PKIX-CommonTypes-2009.asn1
PKIX-X400Address-2009.asn1
PKIX1-PSS-OAEP-Algorithms-2009.asn1
PKIX1Explicit-2009.asn1
PKIX1Explicit88.asn1
PKIX1Implicit-2009.asn1
PKIX1Implicit88.asn1
PKIXAlgs-2009.asn1
PKIXAttributeCertificate-2009.asn1
PKIXCMP-2009.asn1
PKIXCRMF-2009.asn1
Raster-Gr-Coding-Attributes.asn1
Raster-Gr-Presentation-Attributes.asn1
Raster-Gr-Profile-Attributes.asn1
SMIMESymmetricKeyDistribution-2009.asn1
SecureMimeMessageV3dot1-2009.asn1
```

10.6.2 Фірмові і стандартні

Тут розглядаються фірмова бібліотека RPC та стандартна ASN1X компанії SYNRC для бінарної серіалізації внутрішнього бінарного представлення віртуальної машини Erlang та бінарного представлення сімейства розміточних форматів і їх парсерів BER, DER, PER, кодери і декодери яких генеруються з мови специфікацій та протоколів ASN.1.

10.6.2.1 Ericsson Erlang BERT/ETF

Серед фірмових можна відзначити Ericsson Binary Erlang Term Format, який використовується в промисловості більше 10 років починаючи з Github. SYNRC використовує бібліотеку SYNRC RPC яка генерує кодери і декодери (API SDK) для будь яких Erlang структур на наступні мови програмування та мови визначення протоколів:

1) IDL/COM;
2) JavaScript;
3) Google gproto;
4) Apple Swift;
5) Erlang validation.

Завдяки генератору BERT парсерів SYNRC RPC компанія NYNJA змогла уніфіковано працювати з об'єктами системи в двох форматах BERT і GPROTO і залишилася після апробації на BERT.

10.6.2.2 ITU IETF ASN.1 BER/DER/PER Apple/Microsoft/OpenSSL

До стандартних можна віднести бінарні формати для серіалізації сертифікатів та обгорток криптографічних ключів які використовуються в PKI X.509. Завдяки промисловому і повному компілятору ASN.1 у складі Erlang/OTP ми маємо змогу побудувати повністю API SDK інфраструктуру на Erlang мінімальними зусиллями не гублячи сумісність з фірмовим форматом BERT/ETF, так як він все одно використовується в середині віртуальної машини після конвертації. Так була продемонстрована сумісність протоколів CHAT BERT (HRL) та результату ASN.1 компіляції CHAT BER.

1) ASN1C;
2) ASN1CC;
3) ASN1SCC;
4) ASN1SCG (Swift Code Generation);
5) ASN1JCG (Java Code Generation).

В даній статті розказується про один з двох генераторів коду DER/BER/PER кодерів і декодерів SYNRC для мови Swift — ASN1SCG, який генерує код невідмінний від схеми кодування і декодування яку Apple пропонує в свої бібліотеках asn1, crypto та certificates.

10.6.2.3 Google gproto/GRPC

В першу чергу цей формат набув популярності на пристроях Apple, так як протокол основного AP сховища Apple Cassandra використовує цей формат. Значною проблему є переускладеність кодогенератора і інфраструктури компілятора.

10.6.3 Бібліотеки Apple

Починаючи з вести 2020 року, коли Apple анонсувала CryptoKit для X.509 інфраструктури зокрема, компанія випустила наступні бібліотеки, які показується стандартний механізм який Apple передбачає для Swift авторів як вони мусять користуватися PKI X.509 обгортками, сертифікатами, ключами та взагалі будь-якими ASN.1 протоколами.

1) apple/swift-asn1
2) apple/swift-crypto
3) apple/swift-certificates

SYNRC ASN1SCG — це ASN.1 компілятор в мову Swift з використанням схеми кодування Apple, написаний на мові Erlang компанією SYNRC. Тут показуються приклади генерації кодерів і декодерів для деяких структур сімейства PKI X.509 в форматі прикладів Apple (насправді відрізнити конкретно ці неможливо).

10.6.4 Техніка компіляції

Оскільки мова ASN.1 має чіткий синтаксис побудова її компілятора може базуватися на BNF парсер генераторах таких як lexx/yacc або їх аналонах на інших мовах програмування leex/yexx (Erlang), Citron (Swift), ANTLR (Java). Загалом нас цікавитимуть LL, LR, LALR, SLR. Але найшвидші парсери це потокові бінарні, Erlang-овий саме такий і займає 2000 рядків разом з токенайзером на 100 рядків.

Технічно, ASN.1 компілятори є AST транформаціями з вихідної мови в цільову, в даному випадку з Erlang шляхом Elixir в Swift. Основне завдання при цьому побудувати базовий розміточний парсер у вигляді мікро-бібліотеки, яку буде використовувати як хелпери згенерований код. В нашому випадку такою бібліотекою є swift-asn1 від Apple тому це теж нам полегшує завдання. Таким

чином наш згенерований код сумісний з усіма екстеншинами з бібліотек Apple.

10.6.4.1 Розміточний парсер

В нас в ЗОШ №5 вчився математик Сергій Книш, який закінчив МГУ в 19 і поїхав викладати в Каліфорнію. Я коли був в Сан-Франціско заходив до нього в гості і він розказува таку байку, а він на байки багатий бо він писав мережевий протокол IPX для шкільного комплексу Корветів і БК-0010 шоб можна було в іграшки по мережі гратися і міг в пам яті сумувати ряди з точністю до всіх розрядів калькулятора. Так от він розказував шо коли ше тільки починався C++, то таблицю віртуальних методів використовували як масив який індексувався байткодом операції, і коли ви читали байт зі стріма ви простоо викликали функцію з цим індексом в таблиці віртуальнихх методів для десеріалізації цього типу даних. Хоча це байка але в ній є зерно істини, розрізати десеріалізатор і серіалізатор на примітивні функції, кожна з яких займається своїм байт-кодом — техніка, що притаманна багатьом бібліотечним парсерам на мовах програмування C++, Rust, Swift, Java, тощо. Приблизно така сама архітектура і у розміточного TLV парсера Apple.

10.6.4.2 Послідовності та множини

10.6.4.3 Рекурсивні суми та їх теги

Перше шо ви повинні зробити перед тим як публікувати ASN.1 компілятор на сайті ITU.INT це пересвідчитися шо він сприймає класичне визначення рекурсивного списку з книжки Олів є Дебюсона [6].

10.6.4.4 Імплісіти, експлісіти та опшинали

Якщо ви хочете позначити поле додатковою інформацією такою як можливіть приймати значення NULL (OPTIONAL) та вид тегування (129, 160).

10.6.4.5 Переліки та цілочисельні переліки

10.6.4.6 Тензори

10.6.5 Висновки

Найкращий спосіб вивчити ASN.1 — це написати ASN.1 компілятор. Був детально проаналізований X.680 стандарт та знайдені конкуретні переваги над іншими компіляторами: 1) більшість не підтримують рекурсивні типи даних, 2) більшість не підримують повністю всі ASN.1 файли ITU.INT, 3) більшість не підтримують тензори. Наш компілятор зроблений таким, що усуває ці недоліки. Що стосується швидкості парсерів Apple для мови Swift, то вони в середньому в два рази повільніші за fast + +21.

Реалізація використовує потоковий бінарний парсер ASN.1 файлів на 2000 рядків коду разом з токенайзером, що йдуть в дистрибутиві Erlang/OTP в аплікейшині asn1, а сам компілятор виконаний як script файл для мови Elixir на 600 рядків і не потребує компіляції. Найближчі конкуренти asn1c і asn1icc займають 25000 рядків.

Доведення коректності компілятора має комбінаторний вигляд, а головна теорема стверджує шо за 2 проходи, можна повністю скомпілювати ASN.1 X.680 стандарт в будь яку мову програмування, кожен прохід містить доведення коректності 11 циклів, разом які використовують 50 розгалужень кожне з яких доводиться окремо в комбінаторному стилі кейс аналізом. За перший прохід будують часткові форвард декларації, які повністю стають насиченими на другому проході. В процесі першого проходу не зберігаються файли, зате між проходами не очищується контекст, який містить три типи ключів: визначення типів для носіння полів та їх властивостей, синоніми, та специфікатори тензорів. Також компілятор містить verbose опцію -v яка показує всі стадії насичення контексту.

На розробку компілятора був витрачений 1 людино-місяць та згенеровано ним 30000 рядків Swift 5.8 коду який працює на Windows/Linux/Mac. Цей код покриває всі конверти які можна знайти на сайті ITU.INT у вигляді ASN.1 файлів. Сюди входять PKIX, CMP, CMC, LDAP, CMS, PKCS, X.400, тощо. Компілятор asn1scc написаний на F який підтримується Європейським Космічним Агенством не працює на повній множині всіх цих декларацій, а займає теж немало, 32000 рядків на F.

10.7 Протокол розмежування доступу ABAC

10.7.1 PEP

10.7.2 PIP

10.7.3 PDP

Розділ 11

Апробація

ЄРЗ (Єдиний реєстр зброї НПУ), СУСЗЦЗ (Система управління силами та засобами цивільного захисту ДСНС), ЄІС (Єдина інформаційна система МВС), ФП МТРЗ (Функціональна підсистема матеріально-технічного та ресурсного забезпечення МВС), ГСЦ (Головний сервісний центр МВС).

Розділ 12

Висновки

Список використаних джерел