Системи Електронного Урядування

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 «Системи Електронного Урядування»

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: FIPS 199, FIPS 200, 800-37, 800-53, 800-53A, 800-108r1, 800-162.

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

Постійне посилання твору: https://erp.uno/book/erp.pdf
Видавець: ТОВ "Системи Електронного Урядування"

ISBN — 978-617-8027-23-0

Підготовлено до друку на Подолі, м. Київ.

Зміст

Передмова автора

Присвята

У колофоні зазначено, що цей посібник присвячено всім державним службовцям України. Це означає, що як невід'ємна частина соціоінформаційної системи державного устрою вони є основними кінцевими користувачами нашої платформи. Саме для них — майбутніх розробників, операторів, архітекторів, керівників та аналітиків — і написано цей твір. Їхня щоденна праця забезпечує стабільність та розвиток державних інституцій, а впровадження новітніх технологічних рішень має на меті полегшити й оптимізувати цей надзвичайно важливий процес.

Ідея виникнення та історія розвитку системи

Ідея розробки ERP/1 зародилася одразу після того, як я став на шлях підприємництва у 2005 році. Тоді я стояв перед вибором платформи для побудови моделі, котру планував впроваджувати, досліджуючи OCaml, Haskell і Erlang. Зрештою, моїм критеріям мотивації та технічним вимогам найкраще відповідала саме екосистема Erlang. Так розпочався мій професійний шлях: спершу був проєкт турецької соціальної мережі, згодом — робота у ПриватБанку, створення архітектури для месенджера NYNJA, навчання в аспірантурі з розробкою AXIO/1 і, нарешті, масштабні державні та міністерські системи.

Окремі частини системи, зокрема клієнтські компоненти для TWAIN-сканування та взаємодії з документами Word і Excel, було написано мовами C# та F#, оскільки мій найперший професійний досвід був тісно пов'язаний зі стеком .NET. Ось уже понад 18 років ERP/1 успішно обслуговує потреби замовників різного рівня, а фундаментальні ідеї платформи за цей час поширилися на екосистеми багатьох сучасних технологій, таких як LiveView, HTMX, Turbo, і навіть вплинули на такі старіші вебфреймворки, як Ocsigen, WebSharper, Lift та Nitrogen.

Сьогодні ERP/1 на базі Elixir повністю охоплює всі технологічні рівні державних інформаційних систем. При цьому вартість воло-

діння платформою — що є одним із головних показників якості — зведено до абсолютного мінімуму. Це стало можливим завдяки тому, що всю систему написано єдиною мовою програмування і вона здатна повноцінно функціонувати на одному вузлі в межах однієї віртуальної машини. Для порівняння: зібраний код кожного з базових модулів ERP/1 легко вміщується на одній дискеті формату 3.5" ємністю 2.88 МБ.

Ефективність подібних підходів та низьку вартість володіння ними давно довели відомі продуктові компанії: WhatsApp, ПриватБанк, RabbitMQ, Basho, AdRoll, розробники Crytek Warface. Яскравим прикладом є компанія WhatsApp: на момент продажу гіганту Facebook її інфраструктуру обслуговували лише 6 інженерів, а вартість компанії сягала 19 мільярдів доларів. Системи на базі ERP/1 побудовано з використанням тієї ж самої надійної виробничої технології.

Розкриття компонентів основної мотивації

Під час розробки системи ми послідовно застосовували філософію мінімалізму в галузі програмного забезпечення[1], про яку я вперше детально розповідав на одному з київських технологічних форумів ще 2013 року. Якщо викласти її суть стисло, вона зводиться до низки оптимізаційних критеріїв у вимірах часу, якості, людського та ресурсного капіталу:

1) зменшення часу виконання та розробки поряд зі значним подовженням життєвого циклу системи і підвищенням її ефективності. Основні метрики: Computation/Time, Weeks/Release, Feature/Hour, Hour/Lifetime; 2) подолання надмірної складності та зниження вартості розробки разом зі збільшенням надійності та концептуальної простоти. Основні метрики: Bugs/Code, Failures/Period, Cost/Support, Time/Fix; 3) усунення невизначеності на користь максимальної зрозумілості, відкритості та передбачуваності коду. Основні метрики: Committers/App, Issues/Feature, Msgs/Issue, Commits/User; 4) скорочення витрат на інфраструктурне обслуговування і додаткові ресурси задля значного збільшення обсягів обробки даних і обчислювальної спроможності. Основні метрики: Cost/Information, Machines/User, Data/User, Size/Features, Requests/Time, Information/User.

Якщо звести всі ці показники до єдиної всеосяжної максими, вона звучатиме так: ідеальний код — це той, котрий найлегше верифікувати (візуально, інструментально чи суворо математично), що, у свою чергу, неодмінно зумовлює його фундаментальну мінімалістичність. Наприклад, повна формальна верифікація рівня Г7 згідно з класифікатором ТЗІ (технічного захисту інформації) є недосяжною

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

для таких систем, як типові SQL-бази даних. Адже практично неможливо математично довести всі без винятку властивості закритих пропрієтарних платформ на кшталт Oracle чи навіть потужних, але вкрай складних відкритих продуктів зразка PostgreSQL. Тому для підтвердження того, що система зберігає свої задані властивості, інженерам доводиться застосовувати безпосередній контроль типів і стовпців у таблицях, а всі функції згортки (рекурсори й фолди) власноруч писати в коді, наче це роблять курсорні генератори запитів у SQL. Тим не менш, до стандартного арсеналу Ericsson/OTP входить розподілена реляційна база даних Mnesia. І хоча вона теж не позбавлена певної академічної складності, її дизайн елегантно обмежений простим інтерфейсом, що гармонійно вписується в парадигму типізації System F.

Коментарі та настанови для слухачів курсу

Цей навчальний курс є унікальним передусім тому, що предмет нашого дослідження — інформаційна система ERP/1 — відзначається продуманою формалізацією і чудово підходить як дидактичний матеріал для навчання фахівців. Разом із цим, ця ж система може похвалитися багатою історією успішних промислових впроваджень і безкомпромісною підтримкою сумісності.

Курс логічно поділено на 12 частин, кожна з яких послідовно розкриває певний етап розробки програмного забезпечення відповідно до стандартів ISO-9001. Матеріал охоплює повний цикл вітчизняного виробництва: від системного аналізу предметної області та роботи зі складною нормативно-юридичною документацією, через формування високорівневої архітектури, створення ескізного проєкту (з оглядкою на модель OSI та парадигму Закмана), — і аж до низькорівневих нюансів. До таких належать, зокрема, внутрішні механізми перевірки кваліфікованого електронного підпису (КЕП) та симетричного шифрування за алгоритмами ДСТУ-4145, реалізовані повністю автономно без залучення зовнішніх допоміжних криптографічних бібліотек. Завдяки високій щільності інформації та прикладній спрямованості цей посібник може стати вкрай корисним підручником і для фахівців напряму електронного урядування та цифрової трансформації державного апарату.

Матеріал створено з суворим урахуванням жорстких вимог, що висуваються до архітектури EDGE-офісів. Тут розглядається робота з системами, яким притаманні підвищені пороги допуску щодо розміру компільованої кодової бази, затримок мережевої взаємодії (latency), стійкості до відмов, рівня розподіленості, вартості подальшої підтримки, обчислювальної ефективності, багатства функціональності та експлуатаційної відповідності актуальним стандартам.

Подяки

Я хотів би висловити найщирішу вдячність усім своїм вчителям, наставникам та академічним колегам, чия праця часто залишається недооціненою, адже у глобальному масштабі інститут педагогіки переживає відчутну кризу поваги до професії викладача. Особлива подяка викладачам математики, інженерії та філософії, які свого часу прищепили мені фундаментальне прагнення до логічної чистоти та краси архітектурних рішень. Крім того, неабияк дякую всім контриб'юторам платформи ERP/1, які невтомно вдосконалювали цей продукт, а також своїм рідним і близьким за їхню багаторічну безмежну підтримку.

Розділ 1

Вступ

Цей посібник докладно описує формальну модель та програмну архітектуру сучасного функціонального верифікованого мовного забезпечення. Його призначення — слугувати фундаментом для побудови складних інфраструктурних процесінгових систем, орієнтованих передусім на державну модель управління процесами. Платформа створена з прицілом на можливість прозорого проведення глибокого зовнішнього аудиту; вона сумісна з різноманітними видами розподілених сховищ, сучасними телекомунікаційними та реєстровими фреймворками, базовими інтернет-утворюючими сервісами, і головне — ідеально підлаштована під завдання автоматизації захищених автономних офісів та великих державних підприємств України у цілому.

Система управління державними підприємствами ERP/1, що презентується в цьому посібнику, є не тільки ідіоматичним зразком побудови сучасних інформаційних державних та комерційних екосистем. Вона насамперед визначає сувору формальну специфікацію та демонструє її надійну імплементацію (що налічує вже безліч реальних національних впроваджень) для потреб інноваційних, високооптимізованих підприємств, які вимагають максимально точних інструментів контролю операцій і беззаперечних гарантій цілісності даних.

Телекомунікаційна платформа Erlang/OTP, створена в надрах компанії Ericsson, успішно та безперебійно застосовується у світовій індустрії передовими мобільними операторами вже понад 30 років, а її віртуальна машина і досі заслужено вважається однією з найнадійніших у галузі. Інформаційні системи управління підприємствами на її базі вже багато років використовуються не лише в телекомі, але й в елітарній банківській сфері, високошвидкісному фінансовому процесінгу транзакцій, стійких розподілених брокерах повідомлень та в IoT-секторі. Ви в будь-який час можете переглянути роботу наших демо-модулів системи ERP/1, розгорнутих у спеціальному захищеному середовищі з власним автономним центром генерації та випуску ECC X.509 сертифікатів. На сторінках цієї книги ви знайдете

вичерпний перелік ключових модулів системи та опис фундаментальних сутностей схеми.

В основі нашої концепції лежить ідея створення універсальної базової платформи для розробки та стабільного функціонування різноманітних інформаційних реєстрів і розподілених баз (чи банків) даних будь-яких масштабів: від невеликих міжсистемних довідників і спеціалізованих класифікаторів до високонавантажених корпоративних, місцевих та масштабних державних інформаційних ресурсів. Переконаний, що цей посібник буде вкрай корисним для всіх, хто бажає глибоко зрозуміти принципи роботи, архітектуру та методики створення надійних інформаційних платформ, які сьогодні активно застосовуються як у вітчизняному державному, так і в комерційному секторах.

Розділ **2**

Національна програма інформатизації

Мета Національної програми інформатизації (НПІ) полягає у створенні єдиного цифрового простору як проміжного етапу розбудови зрілого інформаційного суспільства. Це прозоре правове середовище, яке надійно захищене, базується на передових міжнародних стандартах, повноцінно забезпечує інформаційні потреби та реалізує права громадян на доступ до достовірної інформації, істотно підвищуючи при цьому ефективність державного управління.

Національна програма інформатизації бере свій початок від Закону України № 74/98-ВР 1998 року та продовжує розвиток у чинному Законі 2024 року № 2807-IX. Головним чином НПІ закріплює протоколи ініціації, контролю та завершення програм інформатизації як на державному, так і на місцевому рівнях. Вона чітко визначає ключових суб'єктів: генерального замовника (Міністерство цифрової трансформації), керівників проєктів, виконавців та спеціалізованих підрядників. Крім того, Програма регламентує життєвий цикл розробки програмного та апаратного забезпечення згідно зі стандартами ISO/IEC 12207 та ISO 9001, а процеси супроводу — згідно з ISO/IEC/IEEE 14764:2022.

Основне завдання НПІ — безперервна оптимізація підприємств та установ у структурі органів виконавчої влади (ОВВ). Для досягнення цієї мети система державного управління повинна мати дієві інструменти інституційної самотрансформації та забезпечувати узгоджену архітектуру IT-комплексів.

2.1 Структурне ядро ОВВ

На мета-рівні безперервний процес реформування органів виконавчої влади (ОВВ) спирається на три фундаментальні складові. У структурі кожного міністерства як структурне ядро сучасного цифрового відомства мають бути наявні:

1. **Управління юстиції та внутрішнього контролю** — необхідний атрибут для забезпечення незалежного внутрішнього аудиту, комплаєнсу та службових розслідувань.

2. **Департамент інформаційних систем (IT-департамент)** — виступає як окрема юридична особа з власним кодом ЄДРПОУ, що виконує роль продуктового власника (Product Owner) для цифрових рішень і відповідає за цілісність державних реєстрів.

3. **Управління інституційної трансформації** — підрозділ, що автоматизує та оптимізує організаційні бізнес-процеси за допомогою впровадження IT-продуктів.

Решта профільних управлінь додаються залежно від специфіки міністерства, але ці три стовпи разом із патронатною службою є обов'язковим фундаментом. Правоохоронні та контролюючі органи (НАЗК, НАБУ, Мін'юст тощо) повинні мати безпосередній або опосередкований доступ до першого компонента (управління юстиції), що суворо регламентується правилами ABAC (Attribute-Based Access Control).

Зі свого боку, Мінцифра, як координатор усього метапроцесу цифрової трансформації, отримує авторизований доступ до третього компонента (управління трансформації). Архітектура та шини обміну даними між усіма системами електронного документообігу (СЕД) і реєстрами координуються IT-управлінням Мінцифри та екосистемою «Дія».

2.2 Стратегічні завдання цифровізації

Завдання державної цифровізації та інституційної трансформації передбачає виконання таких стратегічних цілей:

- кожне міністерство повинно мати свій автономний і висококомпетентний IT-департамент, який операційно обслуговує відповідні державні реєстри;

- міністерства створюють управління трансформації, що працюють на базі єдиної платформи для зручного моделювання бізнес- та адміністративних процесів (BPMN);

- Мінцифра розробляє та затверджує єдині регламенти роботи для IT-управлінь та управлінь трансформації у масштабах уряду.

Платформа «Дія» при цьому забезпечує не лише базову шину даних та коробкове рішення «Дія: Документообіг», але й виступає провайдером ліцензій для інших учасників ринку (наразі видано понад 30 таких ліцензій).

Оскільки соціоінформаційні системи повинні залишатися максимально автономними та мати керований життєвий цикл, найефективніше реалізовувати IT-департаменти у формі окремих державних підприємств. Цей підхід убезпечує ключові інтелектуальні ресурси від ручного політичного втручання та гарантує стабільний захист персональних даних громадян.

2.3 Модель міжвідомчої взаємодії

Для архітектурного проєктування міжвідомчої взаємодії розглядається базова перелікова модель ОВВ, що формує каркас державної інформаційної системи:

1. Кабінет Міністрів України;
2. Міністерство освіти і науки України;
3. Міністерство охорони здоров'я України;
4. Міністерство оборони України;
5. Міністерство соціальної політики України;
6. Міністерство у справах ветеранів України;
7. Міністерство закордонних справ України;
8. Міністерство юстиції України;
9. Міністерство внутрішніх справ України;
10. Міністерство економіки України;
11. Міністерство енергетики України;
12. Міністерство фінансів України;
13. Міністерство розвитку громад і територій України;
14. Міністерство цифрової трансформації України;
15. Міністерство захисту довкілля та природних ресурсів України.

2.4 Протокол державної інституційної трансформації

Для ефективного управління структурними змінами в режимі реального часу пропонується впровадити протокол **«ОВВ 2.0»**. Він є розширенням базового протоколу системи електронного документообігу (згідно з постановою КМУ №55). Його формальним втіленням стає протокол **ДІТ** (Державна інституційна трансформація).

Цей спеціалізований інформаційний протокол визначає такі базові операції над організаційною структурою відомства та її політиками:

1. Створення, злиття та ліквідація структурних підрозділів;
2. Розробка, погодження та модифікація установчих і конституційних документів;
3. Проєктування технічних завдань для розгортання інформаційних систем підрозділів;
4. Збір вимог, вибір та подальше впровадження відповідних програмних продуктів;
5. Запуск, супровід та моніторинг операційної діяльності структурних підрозділів.

У процесі функціонування як класичного операційного документообігу, так і інституційного протоколу ДІТ формуються криптографічні зліпки кваліфікованих електронних підписів (КЕП) відповідальних посадових осіб. Ці незмінні сліди дозволяють здійснювати надійний аудит системи — вони перевіряються внутрішніми контролюючими органами (управліннями юстиції ОВВ) та безпосередньо координуючим органом — Міністерством юстиції України.

Розділ **3**

Парламент

Розділ 4

Органи судової влади

Судова влада в Україні реалізується через систему судів загальної юрисдикції та Конституційний Суд. Побудова КСЗІ для органів судової влади вимагає врахування специфічної ієрархічної структури судів, органів суддівського врядування та самоврядування, а також установ, що забезпечують діяльність судів. Єдиною інформаційно-комунікаційною платформою судової гілки влади є **ЄСІКС** — Єдина Судова Інформаційно-Комунікаційна Система, яка складається з 10 розділів (2 томи технічних вимог) і охоплює весь спектр судочинних, адміністративних та аналітичних процесів.

Структурні підрозділи

1. Верховний Суд
    - Велика Палата Верховного Суду
    - Касаційний адміністративний суд
    - Касаційний господарський суд
    - Касаційний кримінальний суд
    - Касаційний цивільний суд
2. Вищі спеціалізовані суди
    - Вищий суд з питань інтелектуальної власності
    - Вищий антикорупційний суд
    - Спеціалізований окружний адміністративний суд
    - Спеціалізований апеляційний адміністративний суд
3. Апеляційні суди
4. Місцеві суди (загальні, господарські, окружні адміністративні)
5. Органи суддівського врядування
    - Вища рада правосуддя (ВРП)
    - Вища кваліфікаційна комісія суддів України (ВККСУ)
6. Органи суддівського самоврядування
    - Рада суддів України
    - Громадська рада доброчесності (ГРД)
    - Громадська рада міжнародних експертів (ГРМЕ)
7. Установи забезпечення діяльності судів
    - Державна судова адміністрація (ДСА)
    - Територіальні управління ДСА, допоміжні установи
    - Національна школа суддів України
    - Служба судової охорони

4.1 Ієрархія OID профілів безпеки судової системи

Кожен орган та рівень судової системи отримує унікальний ідентифікатор об'єкта (OID) в гілці `1.2.804.3.1.2`, зареєстрованій у національному дереві OID України (`iso.member-body.ua`). Цей OID вбудовується в X.509 сертифікат як розширення Certificate Policy (`id-ce-certificatePolicies`), що дозволяє апаратно ідентифікувати рівень безпеки та інституційну приналежність кожного суб'єкта системи. Імплементація виконана в модулі `CA.CP` бібліотеки `synrc/ca`.

4.1.1 Базовий та галузевий профілі

OID	Профіль
`1.2.804.3.1.2.1`	Базовий профіль безпеки (Level 1)
`1.2.804.3.1.2.2`	Галузевий профіль безпеки (Level 2)
`1.2.804.3.1.2.3`	Цільовий профіль безпеки ЦОД (Level 3)
`1.2.804.3.1.2.4`	Цільовий профіль безпеки Судів (Level 3)

Базовий профіль визначає мінімально необхідний набір контролів для обробки відкритої та конфіденційної інформації в судовій системі. Галузевий профіль розширює базовий набір контролів відповідно до специфіки обробки чутливої інформації в державних реєстрах та судах, враховуючи вимоги високої доступності й цілісності даних.

Мінімальна кількість цільових профілів безпеки Level 3, необхідних для функціонування будь-якого суду, — **два**:

1. **Цільовий профіль ЦОД** — профіль безпеки центру обробки даних, який базується на галузевому профілі Міністерства цифрової трансформації України (DIGITAL UKRAINIAN GOVERNMENT INDUSTRY PROFILE). Визначає контролі фізичної безпеки, мережевої інфраструктури, резервного копіювання та катастрофостійкості серверного середовища, на якому розгортається ЄСІКС.

2. **Цільовий профіль COURT** — базовий цільовий профіль для всіх судів. Визначає контролі прикладного рівня: управління доступом (ABAC), криптографічний захист судових документів (X.509, CMS), аудит дій користувачів, цілісність реєстрів та захист персональних даних учасників судових процесів.

Кожен конкретний суд або орган правосуддя може додатково мати власний цільовий профіль Level 3 або Level 4, який успадковує і розширює профіль COURT відповідно до специфіки своєї юрисдикції.

4.1.2 Цільовий профіль вищих судів

OID	Суд
1.2.804.3.1.2.4.1	Цільовий профіль вищих судів
1.2.804.3.1.2.4.1.1	Велика Палата Верховного Суду
1.2.804.3.1.2.4.1.2	Касаційний адміністративний суд
1.2.804.3.1.2.4.1.3	Касаційний господарський суд
1.2.804.3.1.2.4.1.4	Касаційний кримінальний суд
1.2.804.3.1.2.4.1.5	Касаційний цивільний суд

4.1.3 Цільовий профіль вищих спеціалізованих судів

OID	Суд
1.2.804.3.1.2.4.2	Вищі спеціалізовані суди
1.2.804.3.1.2.4.2.1	Вищий суд з питань інтелектуальної власності
1.2.804.3.1.2.4.2.2	Вищий антикорупційний суд
1.2.804.3.1.2.4.2.3	Спеціалізований окружний адміністративний суд
1.2.804.3.1.2.4.2.4	Спеціалізований апеляційний адміністративний суд

4.1.4 Цільовий профіль місцевих та апеляційних судів

OID	Профіль
1.2.804.3.1.2.4.3	Цільовий профіль судів (Level 3)
1.2.804.3.1.2.4.3.1	Місцеві суди (Level 4)
1.2.804.3.1.2.4.3.2	Апеляційні суди (Level 4)

Місцеві суди охоплюють загальні, господарські, окружні та адміністративні суди першої інстанції. Кожна категорія успадковує контролі Level 3 та доповнює їх специфічними вимогами до обробки судової документації відповідної юрисдикції.

4.1.5 Органи та установи системи правосуддя

OID	Орган / Установа
1.2.804.3.1.2.5	Органи та установи правосуддя
1.2.804.3.1.2.5.1	ДСА — Державна судова адміністрація
1.2.804.3.1.2.5.2	ТУ ДСА, допоміжні установи
1.2.804.3.1.2.5.3	Рада суддів України
1.2.804.3.1.2.5.4	ВРП — Вища рада правосуддя
1.2.804.3.1.2.5.5	ВККСУ
1.2.804.3.1.2.5.6	Національна школа суддів України
1.2.804.3.1.2.5.7	ГРД та ГРМЕ — громадські ради
1.2.804.3.1.2.5.8	Служба судової охорони

4.2 Таксономія компонент ЄСІКС

Єдина Судова Інформаційно-Комунікаційна Система (ЄСІКС) поділяється на 10 функціональних розділів, згрупованих у два томи технічних вимог. Нижче наведено повну таксономію з кодифікацією субпродуктів.

4.2.1 I. Інфраструктурні сервіси

1. Інфраструктура безпеки (X.509, EUDI)
2. Рольова модель і система доступів (ABAC, IAM)
3. Оркестрація процесів (BPMN)
4. Словникова підсистема (HL7)
5. Мета сервіси схеми даних (SCHEMA)
6. Конструктор форм і процесів (X-FORMS)

4.2.2 II. Господарська діяльність

1. Бюджетне планування і контроль видатків (BUD)
2. Фінансовий і бухгалтерський облік (FIN)
3. Кадрова система (ACC)
4. Публічний документообіг і сервісне обслуговування (CRM)

4.2.3 III. Інфраструктура документообігу

1. Шаблони документів (DOCX, XLSX)
2. Генерація документів (PDF)
3. Генерація бар- та штрих-кодів (QR)
4. Розпізнавання Tesseract (OCR)
5. Текстуально-голосові трансформації (S2T, T2S)
6. Редактор справ консиліуму (CRDT)

4.2.4 IV. Реєстри

1. Виконавчі документи (EXEC)
2. Судові рішення (ORDER)
3. Правові позиції (POSITION)
4. Дайджести (DIGEST)

5. Фізичні особи (CITIZEN)
6. Адвокати (ESQ)
7. Уповноважені (POA)
8. Кадри судової влади (JUDGE)
9. Суб'єкти господарювання (BIZ)
10. Зовнішні реєстри (TREMBITA)

4.2.5 V. Судопис

1. Справи (DEEDS)
2. Провадження (PROSEC)
3. Судові засідання і їх рішення (COURT)
4. Авторозподіл справ (AUTOCASE)
5. Реєстрово-інтеграційні сервіси (CART)

4.2.6 VI. Суддівство

1. Облік суддівських кадрів (ACC)
2. Відбір суддів (FILTER)
3. Кваліфікаційне тестування (QUALIFY)
4. Досьє на суддю (PROFILE)
5. Верифікація (VERIFY)

4.2.7 VII. Інтелектуальні сервіси

1. Короткий зміст (EXCERPT)
2. Рекомендації (RECOMMEND)
3. Практика (PRACTICE)
4. Адвокат (ATTORNEY)
5. Сутності (ENTITIES)

4.2.8 VIII. Бізнес, аналіз і звітність

1. Календарі та завдання (CAL, TASK)
2. Платежі (PAYMENT)
3. Архів (ARCHIVE)
4. Коментування (COMMENTS)
5. Конференц-зв'язок (VKZ)
6. Журнали (JOURNALS)

4.2.9 IX. Інформаційно-пошукова система (court.gov.ua)

1. Контент (CMS)
2. Форми (X-FORMS)
3. Інтеграція (TREMBITA)
4. Звернення (CITIZENS)

4.2.10 X. Кабінет (cabinet.court.gov.ua)

1. Заяви (APP)
2. Листи (LETTER)
3. Звернення (REQ)
4. Довіреності (POA)
5. Виконавчі документи (EXEC)
6. Провадження (PRO)
7. Судові рішення (ADJ)

4.3 Політики проєкту ЄСІКС

Політики є первинним і незмінним документом будь-якого проєкту. Вони визначають правила до того, як формулюються вимоги чи обираються технології. Основні принципи, що лежать в основі ЄСІКС:

1. Політики є первинними документами; всі рішення приймаються на їх основі.
2. Технічні вимоги формуються як похідні від політик.
3. Технічне завдання — єдиний документ, де дозволяється вказувати конкретні продукти.
4. Архітектура залишається агностичною до моменту затвердження ТЗ.
5. Технічні вимоги містять перелік сутностей, процесів, прав і привілеїв.
6. Технічне завдання містить BPMN діаграми, ABAC правила, словники, протоколи.
7. Технічна документація визначається стандартами ISO і ДСТУ.

4.4 Вимоги до безпеки ЄСІКС

Вимоги до інформаційної безпеки ЄСІКС повинні відповідати:

- Закону України «Про захист інформації в інформаційно-телекомунікаційних системах»;
- Закону України «Про захист персональних даних»;
- Закону України «Про основні засади забезпечення кібербезпеки України»;
- Правилам забезпечення захисту інформації (Постанова КМУ від 29.03.06 №373);
- НД ТЗІ 3.6-006-24 (гармонізовано з NIST SP 800-53);
- Стандарту OWASP TOP 10 для безпеки веб-застосунків.

Додатково: засвідчення даних здійснюється виключно КЕП у форматі CAdES-X Long; використовуються засоби КЗІ з чинним позитивним експертним висновком Держспецзв'язку; TLS версії не нижче 1.3; захист від SQL-ін'єкцій, XSS, Command Execution.

4.5 Таксономія системних компонент

Системні компоненти ЄСІКС класифікуються за функціональними категоріями:

- **Data Storages:** MongoDB, SQL, S3, Cassandra, Riak, TiKV.
- **PubSub Buses:** Kafka, MQTT (ISO), AMQP (ISO), XMPP (ISO).
- **Application Services:** Anti Fraud, OTP, Auth, Integration Layer, Digital Signature, MCI, ETL.
- **Facade Management:** Traefik / KONG (OSI Layer 4–7).
- **Virtual Machines:** JVM, CLR, BEAM.
- **External Systems:** Трембіта (ДССУ, Дія, ПФУ, Мінюст).

Розділ **5**

Органи виконавчої влади

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

5.1 Міністерство освіти і науки України

Цей підрозділ є статтею-дослідженням таксономії структури Міністерства освіти і науки як з погляду інформаційної автоматизованої системи, так і в розрізі соціальної структури державного управління. Як приклад, тут наведено цільову предметну модель, що є оптимізованою модифікацією чинної ієрархічної системи Міністерства освіти і науки України.

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 та 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 й образотворчого мистецтва
        - Національна консерваторія
        - Національна академія мистецтв
        - Національна кінематика
    - Інститут фізики і матеріалознавства
    - 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)

5.2 Міністерство у справах ветеранів України

5.3 Міністерство охорони здоров'я України

Таксономія Міністерства охорони здоров'я базується на послідовній інтеграції існуючих клінічних баз, системи громадського здоров'я та профільних дослідницьких інституцій у єдину організаційно-інформаційну мережу. Ця модель зорієнтована на оперативне забезпечення сталого розвитку галузі, медичної освіти й науки.

Структурні підрозділи

1. Патронатна служба
2. Управління медичної освіти та науки
    - Медичні університети й академії
    - Науково-дослідні інститути (НДІ) системи МОЗ
3. Управління державного санітарного контролю та громадського здоров'я
    - Центр громадського здоров'я (ЦГЗ)
    - Департамент епідеміологічного нагляду
4. Департамент фармаконагляду і медичного ліцензування
5. Управління юстиції
    - Департамент медичної експертизи
    - Департамент внутрішнього аудиту
    - Департамент кадрового забезпечення та юридичної підтримки
6. Управління політиками та структурними підрозділами (агенція)
    - Департамент цифрової трансформації (взаємодія з eHealth та ЕСОЗ)
    - Департамент стратегічного планування

5.4 Міністерство внутрішніх справ України

Архітектура МВС чітко розподіляє управлінські повноваження між своїм центральним апаратом та підзвітними йому центральними органами виконавчої влади (ЦОВВ). Це забезпечує ефективне реагування, незалежний внутрішній контроль і надання сервісних функцій населенню без дублювання повноважень.

Структурні підрозділи

1. Патронатна служба
2. Головний департамент формування політик безпеки
3. Управління координації ЦОВВ
    - Національна поліція України (НПУ)
    - Державна служба України з надзвичайних ситуацій (ДСНС)
    - Державна прикордонна служба України (ДПСУ)
    - Національна гвардія України (НГУ)
    - Державна міграційна служба України (ДМС)
4. Головний сервісний центр (ГСЦ)
5. Управління юстиції
    - Департамент внутрішньої безпеки
    - Департамент аудиту та службових розслідувань
6. Управління політиками та структурними підрозділами (агенція трансформації)

5.5 Міністерство закордонних справ України

Структурна модель МЗС відображає вимоги сучасного протоколу та міжнародного права. Вона оптимізована для швидкого обміну захищеною аналітичною інформацією з дипломатичними представництвами та для підтримки зовнішньополітичного консолідованого курсу.

Структурні підрозділи

1. Патронатна служба
2. Управління дипломатичної освіти та аналітики
    - Дипломатична академія
    - Інститут актуальних міжнародних відносин
3. Територіальні управління
    - Департаменти європейської, євроатлантичної, азійської та американської політики
4. Департамент міжнародних організацій (ООН, НАТО тощо)
5. Управління юстиції
    - Департамент міжнародного права
    - Консульська служба
    - Департамент кадрового аудиту та безпеки
6. Управління політиками та інституційного розвитку

5.6 Міністерство оборони України

Цей підрозділ є дослідженням таксономії структури Міністерства оборони України з погляду як інформаційної автоматизованої системи, так і соціальної структури в розрізі державного управління. Як приклад у статті розглядається конкретна структурна модель, що є адаптованою модифікацією чинної ієрархічної системи відомства. Ця структура підсилена повним спектром інституцій, необхідних для організації безперервного науково-освітнього та технологічно-виробничого циклу функціонування оборонного державного органу виконавчої влади.

Як модель гранулярності використано типову українську державну класифікацію (міністерство, управління, департамент, відділ, сектор). Оскільки ця робота зосереджена насамперед на логіці архітектури процесів, тут значною мірою надається перевага формальним моделям, які потребують мінімальних зусиль для своєї верифікації, моделювання та прогнозування. З огляду на те, що формалізація процесів безпосередньо стосується програмного забезпечення, у пригоді стають міжнародні стандарти телекомунікаційних протоколів. Їх сертифікація своєю чергою залучає університети, науково-дослідні та науково-виробничі інститути. Науково-виробничі інститути тут розглядаються у широкому сенсі — як такі, що можуть функціонувати в симбіозі з комерційними об'єднаннями, міжнародними фундаціями тощо. Для адекватної підтримки автономної діяльності всі ці інституції повинні входити до єдиного арсеналу функціональних можливостей міністерства. Насамперед це стосується повноцінного університету четвертого рівня акредитації, навчальні програми якого мають перегукуватися (наприклад) з пропозиціями ключових кафедр Інституту математики НАНУ (див. Додаток 1).

Фізичне розміщення базової інформаційної інфраструктури розгалуженої національної мережі міністерства та його підрозділів також передбачає автономне забезпечення (модель класу EDGE-офіс) із власними ресурсами охолодження, водо-електро-постачання та засобами посиленої охорони. Інформаційна політика формального моделювання допускає використання сучасних інструментальних мов розвитку, здатних підтримувати сувору машинну верифікацію на рівні комплексів ТЗІ Г7 (з можливостями повної математичної доводжуваності), покладаючись здебільшого на надійні телекомунікаційні протоколи та міжнародні стандарти серії ISO (див. Додаток 5).

Мотивація

Основна мотивація цієї роботи полягає у висвітленні таксономії Міністерства оборони України через призму структурної оптимізації, максимальної інституційної автономності та життєстійкості (sustai-

nability), а також здатності до самовідтворення й ефективної підтримки операційного життєвого циклу міністерства в умовах динамічних викликів.

Структурні підрозділи

1. Патронатна служба
2. Управління освіти і науки (агенція)
    - Університет четвертого рівня акредитації (див. Додаток 1)
    - Науково-дослідні інститути (НДІ)
    - Науково-виробничі інститути
3. Управління медицини (агенція)
    - Клінічні наукові дослідження та лабораторії, НДІ профілактичної та військової медицини (MED)
    - Заклади реабілітації («Пуща-Водиця», «Трускавецький», «Хмільник»)
    - Клінічні та мобільні (4) лікарні, військові госпіталі (14)
    - Медичні служби (5 родів військ), центри тактичної медицини, Командування Медичних сил
    - Інститут медицини (на базі ЗДМУ, ХНМУ, ЛНМУ, ТНМУ)
4. Виробничо-промислове управління (агенція)
    - Конструкторські бюро (КБ), концерн «Укроборонпром», ДАХК «Артем»
    - Департамент економічного моделювання та планування
    - Департамент ресурсного забезпечення (SCM, TMS, WMS)
    - Департамент бюджетування і закупівель (FIN)
    - Департамент будівництва, архітектури та масштабування ліній виробництва
    - Телекомунікаційний департамент інформаційних систем
        - Відділ систем урядування
        - Відділ облікових систем
        - Відділ телекомунікаційних систем
        - Відділ безпекових протоколів інтернет-зв'язку
    - Департамент авіації, авіоніки, аеронавтики та безпілотних літальних апаратів (БПЛА) і їхніх систем
        - Відділ авіації
        - Відділ авіоніки

  - Відділ аеронавтики
  - Відділ безпілотних систем
- Департамент машинобудування (тестування і проектування місцевості, SolidWorks)
- Департамент суднобудування та військово-морської техніки

5. Управління юстиції
   - Відділ експертиз і сертифікації (ISO/IETF)
   - Департамент архівної справи (SCAN)
   - Департамент аудиту та 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 та стратегією
   - Мобілізаційний департамент
   - Департамент військово-навчальних програм
   - Департамент координації сил та засобів оборони (CRM, див. Додаток 2)
     - Командування Сухопутних військ
     - Командування Повітряних сил
     - Командування Військово-Морських сил
     - Командування спеціальних та допоміжних сил (ССО, Морська піхота, РЕБ, РХБЗ, ТРО)
     - Командування кібербезпеки та ДШВ
   - Департамент контролю й оперативного управління DFR (див. Додаток 4)
   - Економічний департамент
     - Відділ матеріально-ресурсного забезпечення (WMS)
     - Відділ бюджетування і закупівель
8. Управління політиками та структурними підрозділами (агенція трансформації)
   - Департамент комунікації та кадрової політики
   - Департамент контролю виконання ключових показників (CRM)
   - Департамент планування структурного переходу (аналіз процесів BPMN)
   - Департамент цифрової та інституційної трансформації

Принципи

Одним із головних принципів, закладених у фундамент Міністерства оборони (МО), є створення повноцінної екосистеми вищої профільної освіти, яка повинна функціонувати відповідно до найвищих міжнародних стандартів. Для задоволення фахових потреб та розвитку працівників і військовослужбовців на всіх рівнях цієї таксономії в основу організації роботи покладено безперервне функціонування відомчого університету четвертого рівня акредитації, чиї спеціальності мають повністю покривати наукові та технологічні запити самого міністерства.

Іншим універсальним і життєво необхідним принципом є принцип чіткого розподілу влади, з якого випливає наявність трьох ключових макрокорпусів МО.

1. **Політичний корпус** (до якого належать головне управління позиційною політикою, управління політиками та переходом, а також патронатна служба). Він передбачає виокремлення спеціальної агенції трансформації, яка ініціює створення та виконує структурну апробацію інституцій міністерства. До цього корпусу також входить головне управління, що визначає стратегію застосування Збройних Сил України та відповідних засобів оборони.

2. **Виконавчий корпус** (управління освіти і науки, медичне управління та надважливе виробничо-промислове управління). Цей корпус гарантує сталий розвиток високоінтелектуальних та ресурсоємних структурних підрозділів і виводить їх під автономний контроль профільних агенцій, що дозволяє їм гнучкіше взаємодіяти із зовнішніми технологічними й медичними екосистемами.

3. **Судовий і правоохоронний корпус** (управління юстиції, галузевий трибунал). Це окремий, функціонально незалежний структурний блок, який реалізує процеси внутрішнього аудиту, комплаєнсу та службових розслідувань всередині соціоінформаційних систем оборонного відомства.

Самоорганізація

Для систематизації процесу організаційного будівництва варто виділити ключові структурні етапи та засади:

- Розподіл влади та мінімізація адміністративної таксономії;
- Нормалізація формальних процесів;
- Незалежний аудит міністерства та процес інституційної трансформації;
- Проєктування архітектури структурних підрозділів;
- Апробація результатів впровадження і циклічність процесу оновлення.

Розподіл влади традиційно залишається головним принципом ефективного державного управління. Контролюючі та ревізійні органи повинні спеціалізуватися винятково на перевірках і розслідуваннях, їхня структура має бути операційно виокремленою. Розвиток політик, регуляція бізнес-процесів управління та збір реквізитної інформації мають перебувати в компетенції політичного блоку, тоді як виконання завдань (наука, освіта, матеріальне виробництво та забезпечення) здійснюють суто виконавчі гілки відомства.

Вимоги до документування процесів і збереження даних повинні формуватися на найвищому технічному рівні. Це означає використання еквіваріантної семантики, спрощення архітектури процесів до нормальних форм, максимальну мінімізацію заплутаних горизонтальних зв'язків. Усі вони мають цілковито відповідати актуальним вимогам постанови КМУ №55. Водночас має бути організовано прозорий процес історіографії та цифрового аудиту діловодства й виробничих ланцюгів відповідно до суворого стандарту управління якістю ISO 19510 (що є важливим для сумісності з аудитом за стандартами НАТО). Запуск автоматизованого діловодства передбачається впроваджувати поступово і гранулярно: починаючи від ключових політико-формуючих центрів та їхніх найменш ресурсоємних проєктів, і далі розширюючи охоплення на всі інші департаменти та управління згідно з затвердженою магістральною стратегією.

5.6.0.1 Патронатна служба

Забезпечує сприяння реалізації стратегічних та політичних цілей Міністра оборони, безпосереднє персональне консультування, а також організаційне, інформаційно-публічне й експертно-аналітичне забезпечення його діяльності.

5.6.0.2 Управління освіти й науки (агенція)

Сучасний науковий потенціал, крім безпосередньо освітнього процесу, активно виділяє як науково-дослідний, так і науково-виробничий напрями. Сфери досліджень повністю диктуються актуальними потребами бойових або організаційних департаментів Міністерства оборони. Ці високотехнологічні компанії й лабораторії повинні знаходитися, як і підпорядкований університет, в єдиній орбіті стратегічного впливу МОУ.

5.6.0.3 Виробничо-промислове управління (агенція)

Концепція передбачає повне дзеркальне дублювання сфер безпосереднього виробництва засобів оборони відповідно до профілю керуючих департаментів, що формують політики ресурсного забезпечення. Оскільки військова промисловість консолідує базові ресурси оборонної індустрії, є методично доцільним тримати всю картину планування та виробництва під єдиною ієрархічною парасолькою міністерства. Наявні сьогодні виробничі хаби й об'єднання, такі як АТ «УОП» (колишній «Укроборонпром») та ДАХК «Артем», пропонується інтегрувати в систему як потужні зовнішні конструкторські бюро в єдиному переліку з профільними приватними комерційними підприємствами, з якими планується вибудовувати гнучку проєктну співпрацю.

5.6.0.4 Управління юстиції

Управління юстиції як незалежна гілка бере на себе функцію неформального обліку сил (розширений відділ кадрів та прав) і юридичного фіксування протокольних дій усередині системи. Далі вони систематично аналізуються антикорупційним відділом та департаментом внутрішніх розслідувань і аудиту. Саме тут зосереджено ключовий штат юридичної експертизи, що обслуговує всю масштабну ієрархію міністерства, а також координує правову взаємодію із міжнародними зовнішніми партнерами, такими як блок НАТО. Також під парасолькою цього управління доцільно тримати технічний департамент як автономний центр емісії криптографічних сертифікатів і ключів управління для всього штату працівників.

5.6.0.5 Головне управління позиційною політикою та стратегією

Це сучасне осмислення спрощених і водночас глобально стратегічних функцій Генерального штабу, який безпосередньо взаємодіє з наявними резервами Сил та засобів оборони. Загальна тактична і стратегічна консоль управління підпорядковує всю логіку збройної місії веденню магістральної позиційної політики: від ефективного

розгортання сил та засобів до просунутого дата-орієнтованого прогнозування еволюції подій. Відповідно, правильна позиційна політика завжди спирається на максимально достовірний локальний і глобальний облік наявних ресурсів.

5.6.0.6 Управління політиками та структурними підрозділами (агенція)

Це формальний мозковий центр адміністративного врядування та інституційного переходу від застарілої структури до будь-якої наперед заданої оптимальної моделі (наприклад, тієї, що описана в цьому документі). Управління займається первинним аудитом чинних бізнес-процесів (їх юридичним і процедурним розтином) та проєктуванням їхньої поетапної трансформації в нові урядові й облікові цифрові системи. Свою діяльність структура тісно координує з телекомунікаційним департаментом, здійснюючи проєктний менеджмент: контролює процес впровадження, проводить апробацію систем і подальше їх калібрування.

Процес системної трансформації раціонально поділено на такі категорії (зворотно пропорційно до швидкоплинності впровадження):

1. Оцифровування (діджиталізація) базових існуючих процесів без кардинальної модифікації їхньої логіки (наприклад, класичне діловодство).

2. Створення архітектури нових функціональних процесів та надійного технологічного забезпечення для «вакантних» або новоутворених інноваційних екосистем, таких як виробничо-промислове і технологічне управління.

3. Планування та стратегічний розвиток нових управлінь (це найдовший за часом процес, що охоплює розробку навчальних програм, побудову наукомістких продуктів та їх всебічну підтримку).

Кожна фаза цього структурного метапроцесу вимагає побудови чіткого життєвого циклу проєкту, включно з продуктом, який лежить у його технологічній основі. До цього циклу обов'язково входить повний комплект супровідної документації: технічне завдання, ескізний проєкт, робочий технічний проєкт і детальна технологічна документація. Управління політиками природно розподіляє сфери компетенції з телекомунікаційним підрозділом і підтримує єдині стандарти документування процесів.

Ключові процеси й завдання управління переходом можна узагальнити так: 1) архівна систематизація процесного виробництва; 2) розробка гнучких протоколів запуску нових структурних підрозділів; 3) розробка сучасних процесів захищеного документообігу; 4)

розробка процесів інтегрованого управління та субординаційної координації; 5) створення стандартів технологічних процесів на основі європейських стандартів сімейства ISO 9001.

Висновки та результати

У статті представлена розширена і запропонована до впровадження таксономія оптимізованого і нормалізованого міністерства. Наша модель створена з урахуванням передового міжнародного та корпоративного досвіду організації структур, підкреслює чіткі принципи прозорого розподілу влади та організації як глобального (стратегічного), так і локального (операційного) контекстів для відомств. Цілі стають легко досяжними, адже організація функцій здійснюється в найпряміший та найефективніший спосіб, різко мінімізуючи складність та кількість горизонтальних протоколів взаємодії всередині самої системи.

У додатках до статті наведено таксономію навчальних та наукових програм для залученого університету, а також детальну ієрархію телекомунікаційного департаменту, серед основних функцій якого є розробка та практичне впровадження новітніх IT-систем ЗСУ і МО. У процесі цього первинного концептуального моделювання ми спробували охопити всі ключові органи системи аж до найвищого рівня гранулярності — департаментів, відділів та секторів, вибудовуючи первинні індекси і їхні внутрішні протоколи взаємодії.

Бібліографія

- ДСТУ 2732-94 Діловодство й архівна справа. Терміни та визначення.
- Наказ МОУ від 26.05.2014 №333 «Інструкція з організації обліку особового складу».
- Наказ МВС від 21.11.2017 №608 «Порядок проведення службового розслідування».
- Постанова КМУ від 26.07.2018 №370 «Про затвердження Інструкції з діловодства».
- Наказ МОУ від 07.04.2017 №124 «Інструкція з діловодства в Міністерстві оборони України».
- Постанова КМУ від 07.10.2015 №393 «Положення про юридичну службу».
- Наказ МОУ від 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...».

5.7 Міністерство юстиції України

Мін'юст у цій архітектурі відіграє роль не лише галузевого регулятора правовідносин, але й центрального верифікаційного вузла для інфраструктурних процесів інших ОВВ. Міністерство виступає головним арбітром у реєстраційних діях та забезпечує дотримання законності на технічному та організаційному рівнях.

Структурні підрозділи

1. Патронатна служба
2. Департамент державної реєстрації та нотаріату
3. Департамент виконання кримінальних покарань
4. Департамент судової роботи та експертного забезпечення
5. Власний орган юстиції та внутрішнього контролю (Генеральна інспекція)
    - Департамент внутрішніх розслідувань і комплаєнсу
    - Департамент аудиту реєстрів
6. Управління цифровізації та структурних підрозділів
    - Адміністрування державних реєстрів юстиції
    - Забезпечення кіберстандарту (спільно з ЦОВВ)

5.8 Міністерство фінансів України

Організаційна модель Міністерства фінансів відповідає за макрофінансову стабільність, податкову політику та бюджетування всіх державних ресурсів. Для ефективного управління вона суворо розмежовує блок аналітики, блок адміністрування надходжень та блок контролю.

Структурні підрозділи

1. Патронатна служба
2. Департамент державного бюджету та фінансового планування
3. Департамент податкової та митної політики
4. Департамент управління державним боргом
5. Управління юстиції
    - Департамент фінансового аудиту
    - Служба державного фінансового моніторингу
    - Юридичний департамент
6. Управління політиками та структурними підрозділами (інтеграційне планування з Казначейством)

5.9 Міністерство економіки України

Архітектура цього міністерства сконцентрована на прогнозуванні макропоказників, регулюванні промислового та аграрного секторів, а також на питаннях захисту вільної конкуренції та залучення інвестицій.

Структурні підрозділи

1. Патронатна служба
2. Департамент макроекономічного прогнозування та аналітики
3. Департамент розвитку реального сектору економіки
4. Департамент регуляції зовнішньоекономічної діяльності
5. Управління юстиції
    - Департамент антимонопольного контролю та нагляду
    - Департамент внутрішнього аудиту
6. Управління політиками та структурними підрозділами
    - Департамент методології закупівель (Prozorro)

5.10 Міністерство енергетики України

В основу структурної моделі Міністерства енергетики покладено розподіл за ключовими сировинними та енергогенеруючими напрямами, з виділенням сильного блоку екологічної безпеки та стратегічної синхронізації з європейськими енергомережами.

Структурні підрозділи

1. Патронатна служба
2. Департамент ядерної енергетики та атомно-промислового комплексу
3. Департамент нафтогазового комплексу
4. Департамент електроенергетичного комплексу та зеленої енергії
5. Управління юстиції
   - Інспекція з енергетичного нагляду
   - Департамент внутрішнього аудиту та комплаєнсу
6. Управління політиками та структурними підрозділами (агенція переходу до ENTSO-E)

5.11 Міністерство соціальної політики України

Структура цього органу сфокусована на соціальному захисті, регулюванні ринку праці та пенсійному забезпеченні. Вона спирається на Єдину інформаційну систему соціальної сфери (ЄІССС) як інструмент діджиталізації соціальних послуг.

Структурні підрозділи

1. Патронатна служба
2. Департамент пенсійного забезпечення та соціального страхування
3. Департамент захисту прав дітей та піклування
4. Департамент праці та зайнятості
5. Управління юстиції (внутрішній контроль, координація інспекції праці)
6. Управління політиками та структурними підрозділами (Трансформація ЄІССС)

5.12 Міністерство регіональної політики України

Таксономія спрямована на ефективне управління магістральними інфраструктурними проєктами, процесами післявоєнного відновлення та завершенням комплексної децентралізації.

Структурні підрозділи

1. Патронатна служба
2. Департамент інфраструктури та житлово-комунального господарства
3. Департамент регіонального розвитку та децентралізації
4. Департамент містобудування, урбаністики та архітектури
5. Управління юстиції
    - Аудит будівельних і транспортних інспекцій
    - Внутрішня безпека
6. Управління політиками та інституційного відновлення (ЄДЕССБ, логістичні системи)

5.13 Міністерство цифрової трансформації України

Мінцифра виступає головним ідеолог-архітектором всієї урядової метасистеми електронного врядування. Структура побудована як гнучка IT-агенція, поділена за продуктовими напрямами (додаток Дія, ШСД) та проєктними екосистемами (Дія.City).

Структурні підрозділи

1. Патронатна служба
2. Департамент розвитку електронних послуг (екосистема «Дія»)
3. Департамент розвитку IT-індустрії (Спеціальний режим «Дія.City»)
4. Департамент розвитку цифрової інфраструктури (широкосмуговий доступ)
5. Управління юстиції
    - Департамент захисту персональних даних та криптографічного супроводу
    - Департамент внутрішнього IT-аудиту
6. Департамент системної інституційної трансформації ОВВ (методична координація IT-директорів усіх міністерств)

5.14 Міністерство захисту довкілля та природних ресурсів України

Архітектура цього відомства забезпечує контроль за раціональним використанням природних ресурсів і впровадженням політик сталого розвитку згідно з екологічними директивами ЄС.

Структурні підрозділи

1. Патронатна служба
2. Департамент управління відходами та екологічної безпеки
3. Департамент лісового та водного господарства
4. Департамент природно-заповідного фонду та надрокористування
5. Управління юстиції
    - Державна екологічна інспекція (наглядовий орган)
    - Департамент внутрішнього аудиту
6. Управління політиками та структурними підрозділами (впровадження екологічних реєстрів)

Розділ **6**

Специфікація та сертифікація

6.1 Законодавча база

6.1.1 Загальні положення

Базова версія «ERP/1: Документи» 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-Рўсґ
[3] https://zakon.rada.gov.ua/laws/show/393/96-Рўсґ
[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-сґ
[8] https://zakon.rada.gov.ua/laws/show/373-2006-Р»

6.1.2 Базова версія «ERP/1: Документи»

Продукт «ERP/1: Документи» 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-P»
[10] https://zakon.rada.gov.ua/laws/show/749-2018-P»
[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-РўСЃ

6.1.3 Розширення та додаткові модулі

Розширення «ERP/1: Судопис і Суддівство»

— 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 «ERP/1: 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 «ERP/1: 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_2?pf3516=5708&skl=10
[22] http://w1.c1.rada.gov.ua/pls/zweb2/webproc4_2?pf3516=5709&skl=10

Розширення «ERP/1: 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.

6.2 Функціональні модулі та вимоги

6.2.1 Призначення та цілі впровадження

6.2.2 Ревізія поточних систем

Розділ 7

Політики і вимоги

7.1 Політики

7.1.1 Загальні принципи

7.2 Технічні вимоги

7.2.1 Перелік технічних вимог наведених в політиках

7.2.2 Повний зміст технічних вимог

7.3 Технічне завдання

Зміст технічного завдання

7.4 Посилання на стандарти ДСТУ

7.5 Функціональні вимоги

7.5.1 Вимоги до процесів системи

7.5.2 Вимоги до викликів API

7.5.3 Вимоги до інтерфейсу користувача

7.5.3.1 Обробка помилок

7.5.3.2 Загальні вимоги

7.6 Нефункціональні вимоги

7.6.1 Вимоги до архітектури

7.6.1.1 Визначення термінів і призначення документа

7.6.1.2 Рівні архітектури компонентів системи

7.6.1.3 Маніфест відповідності архітектурі

7.6.1.4 Таксономія системних компонент

7.6.2 Вимоги до безпеки

7.6.2.1 Загальні вимоги до інформаційної безпеки

7.6.2.2 Технічні вимоги до інформаційної безпеки

7.6.3 Вимоги до потужності та ємності

7.6.3.1 Таблиця характеристик продуктивності

7.6.4 Вимоги до системи логування

7.6.4.1 Загальні вимоги

7.6.4.2 Рівні логування

7.6.4.3 Інформація для вендорів про інструментарії логування

7.6.4.4 Події логування

7.6.4.5 Структура журналу логування

7.6.4.6 Вимоги до структурних логів ELK стеку

7.6.5 Вимоги до контролю якості

7.6.5.1 Загальні вимоги до обліку і контролю якості

7.6.5.2 Вимоги до ручного тестування

7.6.5.3 Вимоги до формату сценаріїв та протоколів тестування

7.6.5.4 Вимоги до плану тестування

7.6.5.5 Вимоги до автоматичного тестування

7.6.5.6 Вимоги до результатів приймального (UAT) тестування

7.6.6 Вимоги до інтерфейсів

7.6.6.1 Загальні вимоги

7.6.6.2 Вимоги до відображення інформації

7.6.6.3 Вимоги до швидкодії додатка

7.6.6.4 Вимоги до технології реалізації додатку

7.6.7 Юридичні вимоги

Розділ 8

Служби ERP/1

8.1 Шаблони документів

8.1.1 Вимоги до сутностей

8.1.1.1 Шаблон DOCX/XLSX

8.1.1.2 Набір змінних

8.1.2 Вимоги до процесів

8.1.2.1 Заповнення шаблону

8.1.2.2 Версіонування шаблонів

8.2 Генерація документів

8.2.1 Вимоги до сутностей

8.2.1.1 Заповнений документ

8.2.1.2 PDF-документ

8.2.2 Вимоги до процесів

8.2.2.1 Генерація PDF

8.2.2.2 Накладання КЕП

8.2.2.3 Архівація

8.3 Генерація бар- та штрих-кодів

8.3.1 Вимоги до сутностей

8.3.1.1 Код

8.3.1.2 Накладання

8.3.2 Вимоги до процесів

8.3.2.1 Генерація коду

8.3.2.2 Накладання на документ

8.3.2.3 Верифікація

8.4 Розпізнавання Tesseract

8.4.1 Вимоги до сутностей

8.4.1.1 Документ PDF

8.4.1.2 Результати OCR

8.4.2 Вимоги до процесів

8.4.2.1 Інгестія документа

8.4.2.2 Виконання OCR

8.4.2.3 Повнотекстовий пошук

8.4.2.4 Отримання підсвічування

8.5 Система сканування

8.5.1 Вимоги до сутностей

8.5.1.1 Профіль сканера

8.5.1.2 Сканований документ

8.5.2 Вимоги до процесів

8.5.2.1 Ініціація сканування

8.5.2.2 Доставка результату

8.6 Багатокористувацький редактор

8.6.1 Вимоги до сутностей

8.6.1.1 Редакційний документ

8.6.1.2 Операція редагування

8.6.2 Вимоги до процесів

8.6.2.1 Синхронізація в реальному часі

8.6.2.2 Історія змін

8.6.2.3 Фіксація версії

8.6.2.4 Транзакційність

8.6.3 Призначення та цілі впровадження

8.7 Інфраструктура безпеки

8.7.1 Провайдери української криптографії

8.7.2 Вимоги до сутностей

8.7.2.1 Директорія (LDAP)

8.7.2.2 Сертифікати (X.509)

8.7.2.3 Користувачі (inetOrgPerson)

8.7.2.4 Організації (organization)

8.7.2.5 Криптографічне повідомлення (CMS X.894)

8.7.2.6 Кваліфікований електронний підпис (ДСТУ 4145)

8.7.2.7 Відкликання і перевірка (OCSP)

8.7.2.8 Мітка часу (TSP)

8.7.2.9 Ключі шифрування та сервісні кредениціали

8.7.3 Вимоги до процесів

8.7.3.1 Видача сертифікату (enroll)

8.7.3.2 Автентифікація сертифікатом (check)

8.7.3.3 Відкликання сертифікату (revoke)

8.7.3.4 Оновлення сертифікату (renew)

8.7.3.5 Підпис і верифікація (sign/verify)

8.7.3.6 Шифрування і розшифрування (encrypt/decrypt)

8.7.3.7 Управління секретами (store/retrieve)

8.7.4 Вимоги до серіалізації і сховища

8.8 Контроль доступу

8.8.1 Вимоги до рольової моделі ABAC

8.8.1.1 Об'єкт доступу

8.8.1.2 Суб'єкт доступу

8.8.1.3 Правило доступу

8.8.2 Вимоги до сутностей ABAC

8.8.2.1 Політики доступу

8.8.3 Вимоги до процесів ABAC

8.8.3.1 Життєвий цикл Політики доступу

8.8.3.2 Точки доступу

8.9 Директорія підприємства

8.9.1 Вимоги до сутностей директорії

8.9.1.1 Адміністратор організацій

8.9.1.2 Адміністратор структурних підрозділів

8.9.1.3 Адміністратор співробітників

8.9.1.4 Адміністратор користувачів

8.9.2 Вимоги до процесів директорії

8.9.2.1 Життєвий цикл Адміністратора

8.9.2.2 Життєвий цикл Організації

8.9.2.3 Життєвий цикл Користувача

8.9.2.4 Життєвий цикл Сесії

8.9.2.5 Життєвий цикл Ролі

8.9.3 Вимоги до рольової моделі структурних одиниць

8.9.3.1 Точки доступу

8.9.4 Вимоги до сховища

8.9.5 Вимоги до серіалізації і валідації

8.10 Оркестрація процесів

8.10.1 Вимоги до сутностей BPMN

8.10.1.1 Документ

8.10.1.2 Процес

8.10.1.3 Монітор

8.10.1.4 Задача користувача (АРМ)

8.10.1.5 Системна задача

8.10.1.6 Перехід

8.10.1.7 Подія

8.10.1.8 Таймаут

8.10.1.9 Асинхронне повідомлення

8.10.2 Вимоги до процесів

8.10.2.1 Створення

8.10.2.2 Зупинення

8.10.2.3 Відновлення

8.10.2.4 Архівація

8.10.2.5 Зміна контексту процесу

8.10.2.6 Перехід на стадію

8.10.3 Вимоги до сховища

8.11 Словникова підсистема

8.11.1 Загальні відомості

8.11.2 Вимоги до сутностей HL7

8.11.2.1 Системноутворюючі довідники

8.11.2.2 Похідні довідники

8.11.2.3 Система імен

8.11.2.4 Відображення довідників

8.11.2.5 Налаштування довідникової системи

8.11.3 Вимоги до процесів

8.11.3.1 Створення попереднього словника

8.11.3.2 Редагування попереднього словника

8.11.3.3 Публікація словника

8.11.3.4 Деактивація словника

8.11.3.5 Архівування словника

8.11.4 Вимоги до сховища

Мета сервіси та конструктор

Реєстр сутностей системи (API)

Бібліотека валідації (JSON Schema)

Конструктор сутностей та форм

Конструктор бізнес процесів

Конструктор словників

Конструктор правил доступу

Конструктор шаблонів профілів користувачів

Розділ 9

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

9.1 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.

9.2 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.

9.3 Технологічний рівень зв'язності людей та пристроїв

9.3.1 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.

9.3.2 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»

9.4 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.

9.5 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 **10**

Юридично-документальний рівень

10.1 Вступ

Відповідно до фреймворку системи, верхній п'ятий рівень визначає BPMN-процеси, згідно з якими здійснюється точне відображення юридичних відносин у вигляді електронного документообігу. Кожен крок такого процесу та всі супровідні документи підписуються особистим ключем КЕП (кваліфікованого електронного підпису) посадової особи. Це створює надійне підґрунтя для проведення аудитів, диспутів та службових розслідувань Міністерством юстиції України, із залученням власного Засвідчувального центру в просторі інтернет-імен. Окрім того, цей рівень системи повністю орієнтований на аналітику та прозору взаємодію з громадянами через Систему електронної взаємодії органів виконавчої влади (СЕВ ОВВ).

Юридично-документальні системи класу ERP/1 будуються на базі надійного сховища з єдиним простором ключів (на кшталт Facebook RocksDB), що здатне працювати через інтерфейси Intel SPDK на швидкісних NVMe дисках, наприклад, у складі таких кластерних сховищ, як CEPH. Архітектура враховує той факт, що обсяг обігу документів на великих державних чи комерційних підприємствах може сягати 1 ТБ на рік.

10.1.1 Види документообігів

10.1.1.1 Постанова №55 КМУ

10.1.1.2 Наказ №124 МОУ

10.1.1.3 Закупівлі

10.1.1.4 Провадження

10.1.1.5 Зброя

10.1.2 Функціональні можливості

10.2 Модулі підприємства

ERP/1 є потужним комплексом інструментальних бібліотек (N2O.DEV) та підсистем додатків (ERP.UNO), який використовує загальну шину повідомлень і спільну розподілену базу даних для побудови швидкісних операційних вітрин.

ERP — Даний модуль обліково-реєстраційного рівня надійно зберігає основну ієрархічну структуру підприємства, її схему, метаінформацію про базові типи даних, а також безпосередньо саму інформацію: записи про персонал, інвентар, компанії-контрагенти та офіси підприємства.

CRM — Система функціонального управління зв'язками з громадськістю та органами виконавчої влади: являє собою базову вітально необхідну реалізацію вимог постанови №55 КМУ.

CART — Система гнучкого управління реєстрами: являє собою реалізацію надійного базового сервера для виконання широкого спектра реєстрових та класифікаторних задач.

10.3 Управління ресурсами

Головним чином інформаційна інфраструктура підприємства складається з активних обчислювальних ресурсів (додатків та сервісів, запущених у шині) та пасивних накопичувальних ресурсів (даних, збережених у розподіленій базі).

Для ефективного управління обчислювальними ресурсами SOA-архітектура як системна модель пропонує асинхронний протокол віддаленого виклику процедур безпосередньо на шинах. Разом з екосистемою N2O можна використовувати MQTT та інші транспортабельні шини, послуговуючись такими протоколами, як TCP та WebSocket. Ці асинхронні комунікації часто називають протоколами реального часу, оскільки функції відправлення повідомлень у них завжди миттєво повертають результат (fire-and-forget). Що ж стосується протоколів для традиційної публікації та доступу до даних, то у цьому контексті може виявитися особливо доречним застосування класичного синхронного протоколу HTTP.

10.4 Архітектура врядувальних CRM-систем

10.4.1 Сторінки

Нижче наведено базовий перелік маршрутів 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
```

10.4.1.1 LDAP

Сторінка безпечної авторизації та ідентифікації користувачів.

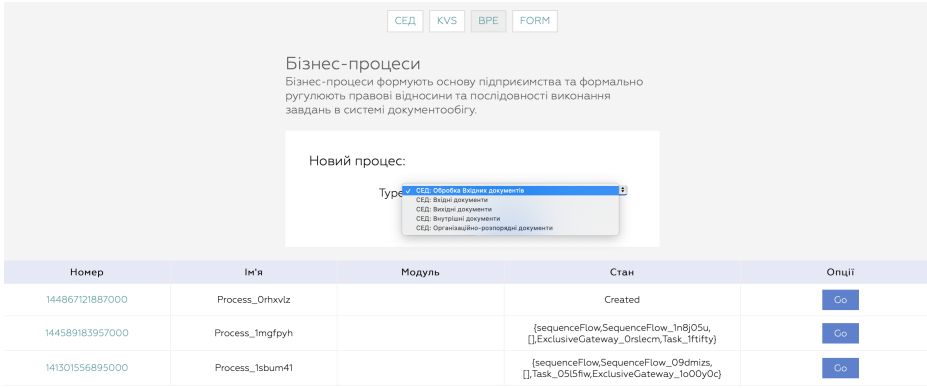


Рис. 10.1 Сторінка авторизації

10.4.2 Комболукап

10.4.3 Сервіси

10.4.4 СЕВ ОВВ

10.4.5 Шаблони

10.4.6 Дерева

10.4.7 Процеси

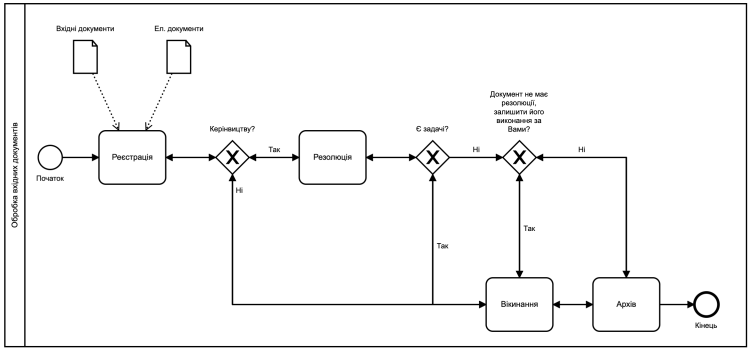
Даний керівний модуль надійно інкапсулює визначення сутностей Схеми, Бізнес-процесів та Форм, які безперервно використовуються у системі Infotech-ERP згідно з академічною методологією фреймворку Закмана.

10.4.7.1 Формування нормативно-довідкової інформації

У цьому модулі автоматизації виділяються такі основні типи процесів для організаційно-розпорядчих документів: «Накази», «Протоколи», «Доручення керівництва».

10.4.7.2 Обробка вхідних документів

Усі вхідні документи автоматично надходять в УДСД (Управління документального забезпечення), ВОРЗГ (Відділ організації роботи зі зверненнями громадян) та ВОДПІ. При надходженні документа уповноважена посадова особа зазначених структурних підрозділів сумлінно реєструє його в системі, після чого стартує процес його подальшої маршрутизації та обробки.



Бізнес-процес обробки вхідних документів

Рис.

Далі зареєстрований вхідний документ надходить або Міністру, Заступнику міністра, чи Державному секретарю для накладання управлінської резолюції, або безпосередньо спрямовується до профільного структурного підрозділу на виконання.

10.4.7.3 Вхідні документи

10.4.7.4 Вихідні документи

Вихідні документи формуються та створюються безпосередньо в підрозділах, які виступають ініціаторами документа. Вони можуть виникати як з власної ініціативи співробітників Міністерства, так і

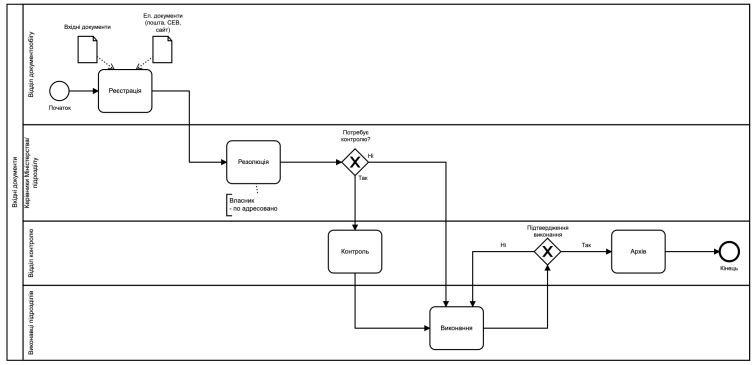


Рис. 10.3 Бізнес-процес обробки вхідних документів

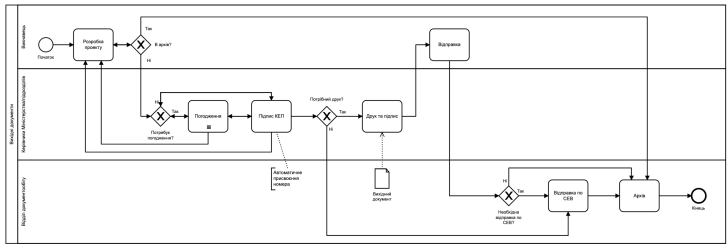
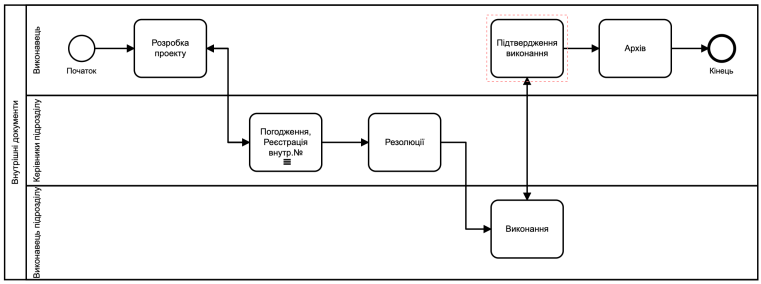


Рис. 10.4 Бізнес-процес підготовки вихідних документів

як прямий результат або відповідь на обробку вхідних документів. Якщо вихідний документ логічно пов'язаний із вхідним документом, то відповідальному працівнику система рекомендує обов'язково вказати семантичне посилання на пов'язаний документ-першоджерело. Регламентна обробка документів виконується за єдиним еталонним уніфікованим бізнес-процесом, незалежно від конкретного місця виникнення документа. Від початку в системі реєструється проект вихідного документа, який неодмінно повинен бути завізований і погоджений із визначеним кваліфікованим переліком погоджуючих посадових осіб. Щойно фінальний підписант накладає свій КЕП, документу автоматично присвоюється вихідний реєстраційний номер, і система ініціює його відправку.

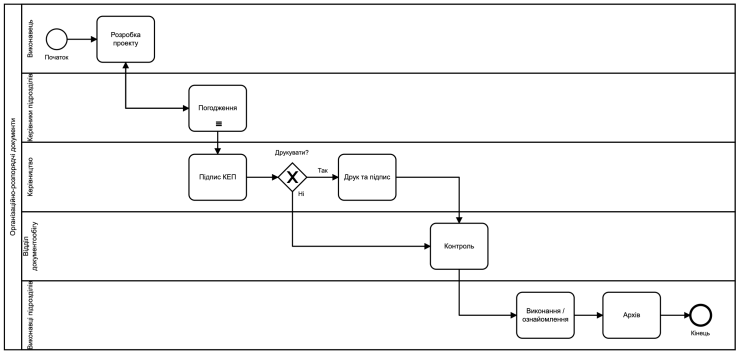
10.4.7.5 Внутрішні документи

Внутрішні документи можуть створюватися та вводитися всіма авторизованими учасниками контуру документообігу. Серед них виділяються такі основні стандартизовані бізнес-процеси внутрішніх документів: «Доповідна записка», «Службовий лист».



Бізнес-процес обігу внутрішніх документів Рис.

10.4.7.6 Організаційно-розпорядчі документи



Бізнес-процес просування організаційно-розпорядчих документів Рис. 10.6

Розробка проєкту документа

На цьому початковому етапі розробляється базовий електронний проєкт документа. Спочатку уважно заповнюються всі необхідні мета-реквізити в електронній картці документа; після її збереження до картки автоматично прикріплюється візуальний шаблон документа, який відіграватиме роль оригіналу. Далі виконавець вносить безпосередній змістовий вміст документа в цей оригінал і зберігає його версію. Ініціатор у системі додає всіх виконавців, кому адресований майбутній наказ. Варто зазначити, що використовуючи поле «Адресовано», система автоматично згенерує та призначить задачі відповідним виконавцям. Тільки після цих процедур документ слід передати на наступний крок.

Погодження (візування)

На даному кроці у суворій послідовності виконується фахове погодження документа усіма особами, які були заздалегідь вказані виконавцем під час створення проєкту. Обов'язковою умовою автоматичної передачі документа на наступний етап є отримання позитивного погодження (візи) від **УСІХ** погоджуючих осіб. В іншому разі просунути документ за маршрутом далі просто неможливо. Якщо принаймні один з візуючих відхилив документ (при цьому він зобов'язаний внести обґрунтований коментар із причинами відхилення і зауваженнями до тексту чи суті наказу) — система повертає документ на першу стадію, інформуючи виконавця про необхідність його доопрацювання. Якщо ж усі без винятку особи погодили проєкт, система автоматично переводить його на наступну стадію. Після остаточного візування є можливість згенерувати та вивантажити офіційний «Аркуш погодження» у вигляді друкованої форми.

Застосування КЕП (кваліфікованого електронного підпису)

На цій критичній стадії документ остаточно підписується в електронному вигляді уповноваженим керівництвом Міністерства. Саме в момент накладання валідного криптографічного КЕП документу присвоюється постійний інвентарний реєстраційний номер. У разі налаштування Системи електронного діловодства (СЕД) на використання візуального QR-коду, він генерується з розмірами 21 на 21 мм і надійно розміщується в нижньому лівому куті першої сторінки документа. Якщо ж політикою СЕД передбачено використання класичного лінійного штрихкоду, його алгоритмічно розміщують у правому кутку нижнього поля першої сторінки оригінального документа.

Підпис у паперовій формі

Після електронного підписання організаційно-розпорядчого документа, у разі суворої необхідності створення так званого «паперового оригіналу», уповноважена особа патронатної служби (помічник Міністра, заступника міністра чи державного секретаря) акуратно роздруковує документ і надає його на «мокрий» підпис безпосередньо керівнику. Якщо ж відповідний організаційно-розпорядчий документ був підписаний локальним керівником певного підрозділу, необхідний паперовий примірник, як правило, роздруковує сам виконавець.

Постановка на контроль

На даному етапі спеціалізований контролюючий структурний підрозділ аналізує завдання за документом і за необхідності здійснює

його постановку на жорсткий контроль, вказуючи також періодичність звітування. Документи та задачі, що перебувають на контролі, завжди доступні уповноваженим особам за окремим фільтром, незалежно від поточного кроку свого опрацювання. Це дає змогу ефективно відстежувати їхній статус на будь-якій стадії життєвого циклу, чітко контролювати своєчасність виконання, проводити комплексну аналітику тощо, не втручаючись напролом в операційний перебіг.

Виконання та ознайомлення

На даному виконавчому етапі відповідальний за ознайомлення або контроль структурний підрозділ бере до уваги директиви, викладені в документі. За всіма документами на контролі в режимі реального часу відстежується їхній актуальний статус. Працівники можуть переглядати, аналізувати та закривати задачі паралельно з тим, як потік процесів продовжує свій рух інформаційними каналами міністерства.

Цифровий шифрований архів

Після виконання завдань та загального ознайомлення адресатів, документ офіційно переходить у довгостроковий цифровий Архів. Система архіву додатково накладає електронні підписи КЕП самого Архіву, надійно цементуючи файл. Крім того, задля гарантування непорушності, Архів навічно утримує повну криптографічну історію та ланцюг усіх застосованих проміжних сертифікатів: починаючи від локального АЦСК та закінчуючи ЦЗО.

10.4.7.7 Звернення громадян

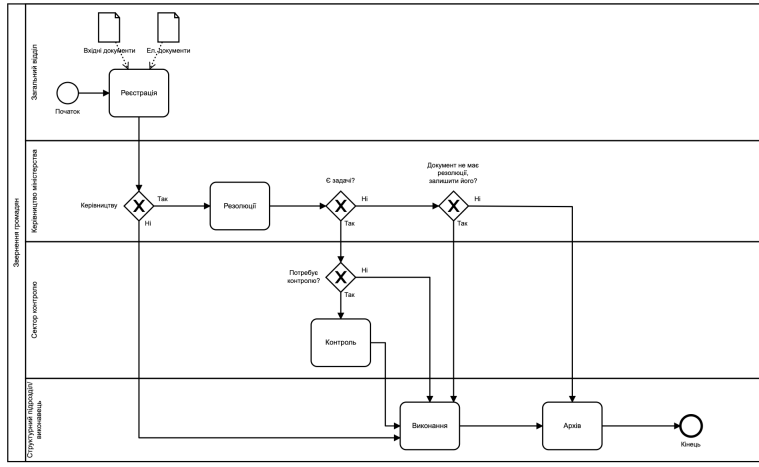


Рис. 10.7 Бізнес-процес опрацювання звернень громадян

10.4.8 Елементи інтерфейсу

У цьому розділі зібрано мінімально необхідну кількість бізнес-форм та UI-компонентів, розроблених спеціально для CRM-сегменту системи електронного діловодства (СЕД). Вони є індивідуальними та критично необхідними для виконання функціональних вимог, висунутих замовниками.

10.4.8.1 Календар

Базовий елемент інтерфейсу «Календар» взято з відкритої бібліотеки NITRO, проте він адаптований і потребує застосування додаткової кастомної стилізації.

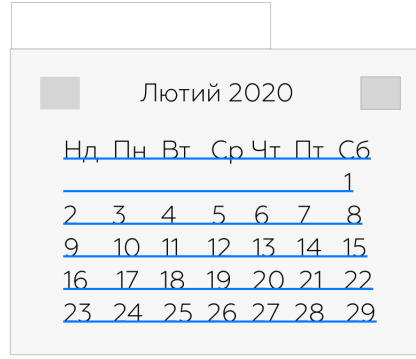
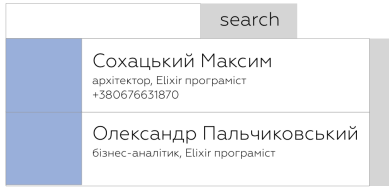


Рис. 10.8 Контрольний елемент «Календар»

10.4.8.2 Пошук за довільними атрибутами

Для забезпечення швидкого та релевантного пошуку за величезними загальнонаціональними словниками та бізнес-об'єктами в системі передбачено створення спеціалізованого скалярного комбо-пошуку. Він дає змогу виконувати віддалені запити за довільними полями в централізованому сховищі даних (наприклад, виконувати миттєвий пошук за довідниками співробітників, індексами населених пунктів КОАТУУ тощо).



Контрольний елемент віддаленого пошуку в базі даних Рис. 10.9

10.4.8.3 Форми редагування та пошуку

Для кожного зареєстрованого типу документа чи сутності в системі створюються дві паралельні форми: форма пошуку і форма редагування (яка одночасно виступає як форма створення нового об'єкта). Наявність цих двох різних сутностей суворо вмотивована відмінністю у логіці їхніх валідаторів. У формі пошуку валідатори апріорі повинні дозволяти залишати поля порожніми, тоді як для форми редагування валідатори зобов'язані безкомпромісно перевіряти сувору повноту та валідність заповнення полів критичних бізнес-об'єктів.

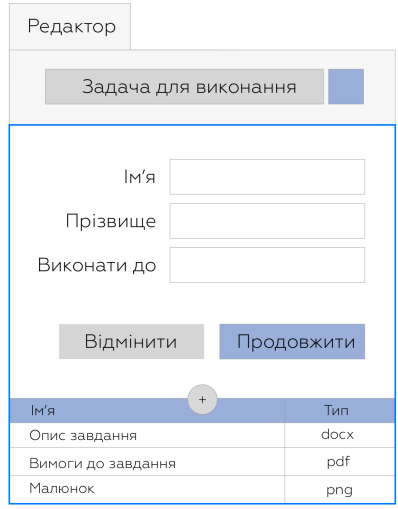


Рис. 10.10 Контрольний елемент інтерактивного редагування документа та прикріплених підлеглих файлів

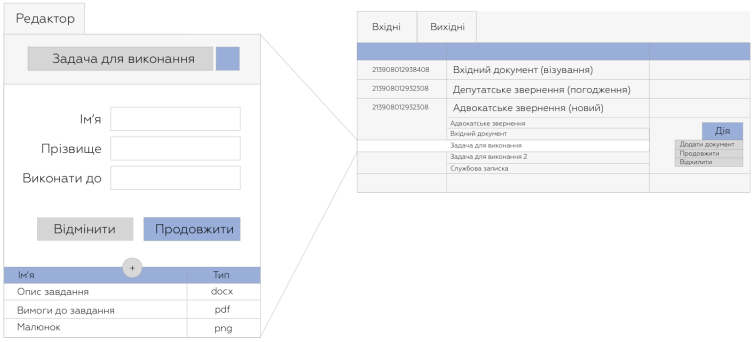
10.4.8.4 Управління бізнес-процесом

Для ефективного управління щоденними завданнями, миттєвого доступу до документів у розрізі певного процесу, створення дочірніх документів, а також для процесів візування, підпису або подальшої маршрутизації документів по бізнес-ланцюгах в інтерфейсі використовується стандартизований контрольний елемент управління бізнес-процесом.

Рис. 10.11 Контрольний елемент управління бізнес-процесами

10.4.8.5 Документи у виконавчих процесах

Під час навігації сторінками та документами окремого процесу інтерфейс гарантує миттєве відображення вмісту будь-якого підлеглого документа у спеціальній лівій робочій панелі головної сторінки користувача, що значно прискорює аналіз та прийняття рішень.



Інструменти навігації за документами обраного бізнес-процесу Рис. 10.12

10.4.8.6 Використання вбудованих контролів у формах

Приклад практичного використання універсального контрольного елемента довільного пошуку-комболукапу всередині складних комплексних форм.

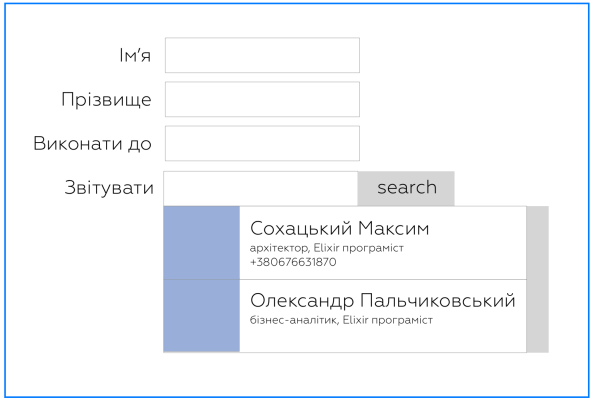


Рис. 10.13 Приклад безшовної інтеграції контрольних елементів у структурованих формах

10.4.8.7 Контрольний елемент КОАТУУ

Наочний приклад роботи спеціалізованого державного контрольного елемента пошуку за класифікатором КОАТУУ.

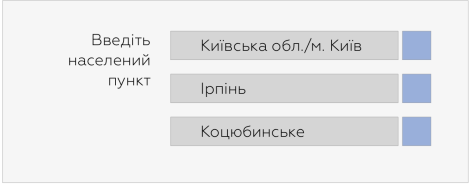


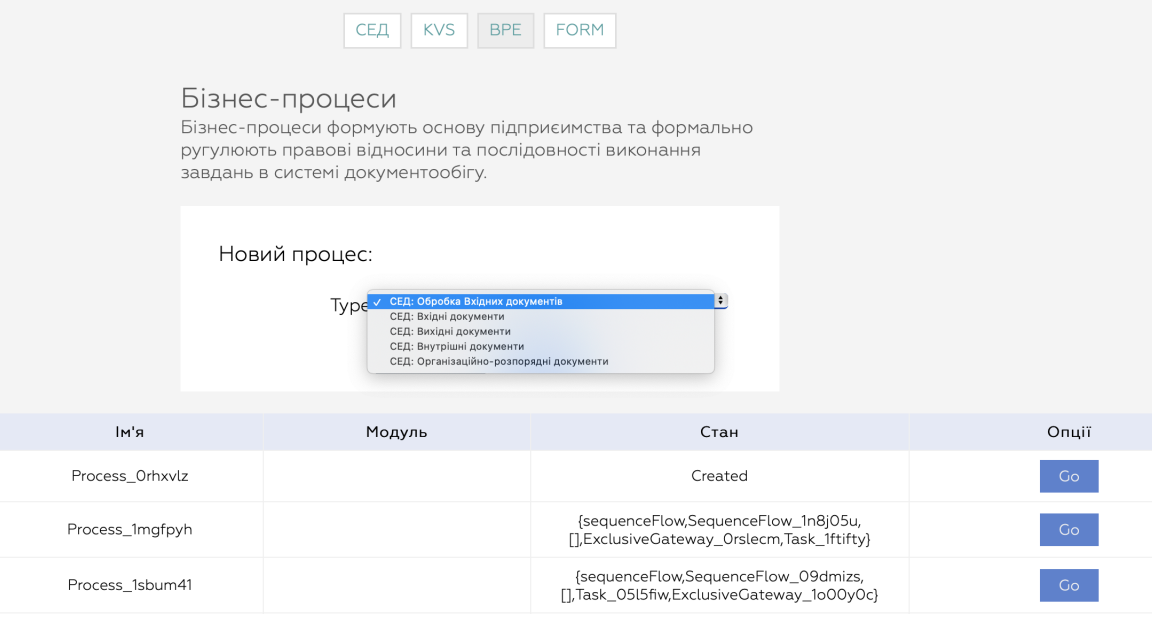
Рис. 10.14 Приклад спеціалізованого використання контрольного елемента КОАТУУ

10.4.9 Редактори та додатки

Тут перелічено контролери основних сторінок, кожна з яких є повноцінним незалежним SPA (Single Page Application) вебдодатком, завантаженим в екосистему.

10.4.9.1 Вхід до системи

Головна сторінка входу та первинної авторизації користувачів з використанням ЕЦП (КЕП).



Захищена сторінка входу в систему

Рис

10.4.9.2 Робота з документами

Основна консоль системи та найголовніша робоча сторінка користувача, яка слугує хабом для безпосередньої обробки документів у робочих бізнес-процесах.

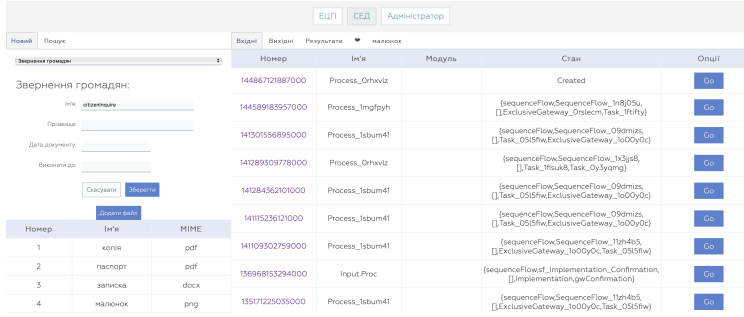


Рис. 10.16 Функціональна сторінка роботи зі вхідними та вихідними документами

При навігації за документами процесу користувачеві доступне миттєве відображення змісту підлеглого документа або його вкладень на лівій інформаційній панелі головної сторінки, що оптимізує час опрацювання.

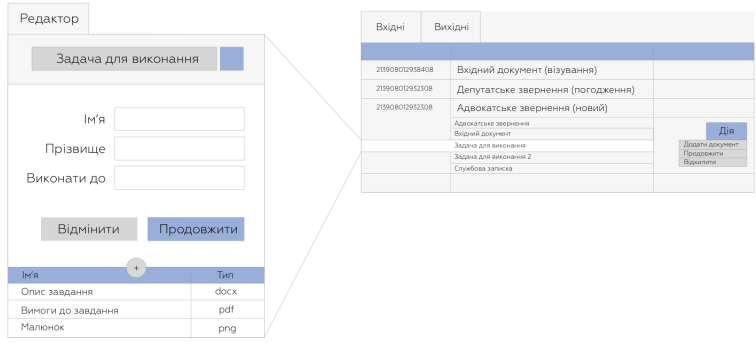


Рис. 10.17 Мініатюри та швидка навігація за підлеглими документами

10.4.10 Конструктор процесів

Тут представлено спеціалізовані адміністративні сторінки управління системою, які призначені для системних архітекторів.

10.4.10.1 Бізнес-об'єкти

Глобальний централізований каталог усіх типів бізнес-об'єктів (документів, організацій, форм), які наразі існують у системі.

10.4.10.2 Бізнес-процеси

Вичерпний перелік усіх зареєстрованих та затверджених бізнес-процесів у системі, із забезпеченою можливістю гнучкого тестування, відлагоджування і проєктування маршрутів.

10.4.10.3 Бізнес-форми

Перелік усіх стандартизованих візуальних форм документів та кастомних бізнес-форм користувача, спеціально зареєстрованих для рендерингу в системі.

10.4.11 Мова програмування FormalTalk

Розділ 11

Обліково-реєстраційний рівень

11.1 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.

11.1.1 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.

11.1.2 Функціональні можливості

11.2 Модулі підприємства

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

WMS — Система управління складом, устаткуванням, деталями.

TMS — Система управління транспортом підприємства.

HL7 — Медична система, яка реалізує міжнародний FHIR стандарт.
МЕДИЧНА КАРТА стаціонарного хворого — №003/о[1]
КАРТА ПАЦІЄНТА, який вибув із стаціонару — №066/о[2]
МЕДИЧНА КАРТА амбулаторного хворого — №025/о[3]
КОНТРОЛЬНА КАРТА диспансерного нагляду — №030/о[4]
МЕДИЧНА КАРТА стоматологічного хворого — №043/о[5]
КАРТА хворого фізіотерапевтичного відділення — №044/о[6]
Медична карта новонародженого — №097/о[7]

[1] https://zakon.rada.gov.ua/laws/show/z0662-12#n2
[2] https://zakon.rada.gov.ua/laws/show/z0668-12#n2
[3] https://zakon.rada.gov.ua/laws/show/z0669-12#n2
[4] https://zakon.rada.gov.ua/laws/show/z0671-12#n2
[5] https://zakon.rada.gov.ua/laws/show/z0678-12#n2
[6] https://zakon.rada.gov.ua/laws/show/z0689-12#n2
[7] https://zakon.rada.gov.ua/laws/show/z0233-16#n7

11.3 Архітектура облікових CART систем

11.3.1 Облік метаінформації

11.3.2 Облік ABAC правил

11.3.3 Облік інфраструктури і само-моніторинг

11.3.4 Облік словників і класифікаторів

11.3.5 Адміністративний облік

11.3.6 Облік таксономії предметної облісті

11.3.7 Облік процесів предметної області

Розділ **12**

Технологічний рівень зв'язності

12.1 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`

12.2 Виробничий процес

12.3 Системи сховищ даних

12.3.1 Реляційні бази даних

12.3.2 Бази даних з єдиним простором ключів

12.3.3 Шини комунікації та брокери повідомлень

12.3.4 Розміщені в пам'яті гарячі дані

12.4 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] }
]
```

12.4.1 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.

12.5 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.

12.6 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`

12.6.1 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]

12.6.2 Приклади

Головні приклади фундації N2O.DEV причасені наступним темам: MQTT та WebSocket дати 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

12.7 Протоколи, схеми та мови їх опису

12.7.1 Мова опису протоколів ASN.1

12.7.2 Мова опису протоколів SOAP/XSD/XML

12.7.3 JSON валідатори draft-07 і JTD

12.8 Формати передачі даних

12.8.1 Бінарні формати ETF/BERT

12.8.2 Бінарні формати DER/BER/PER

12.8.3 Колоночний текстовий формат CSV/CSM

12.8.4 Текстові формати JSON і XML

12.9 Розробка Інтернет додатків

12.9.1 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.

12.9.2 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.

12.9.3 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 їй присвячений друга версія підручника.

12.9.4 Інтерфейс NITRO

12.9.5 Сховище KVS

12.9.6 Логіка BPMN

12.9.7 Додатки MQTT та WebSocket

Розділ **13**

Генерація, валідація і верифікація

Комплексна інфраструктура рівня ERP вимагає бездоганного теоретичного фундаменту для забезпечення безвідмовної роботи алгоритмів. В рамках архітектурного підходу (відомого як *Volume V: Verification* або *Computational Layer*) ми визначаємо набір математичних та інженерних дисциплін. Програмне забезпечення повинно бути не лише емпірично протестованим, але й строго доведеним за допомогою формальних методів. Розділ закладає теоретичний фундамент, оглядаючи сучасний стан формальної верифікації та визначаючи місце пропонованої уніфікованої системи формальних мов у цій галузі.

Формальна верифікація як дисципліна бере початок із робіт Ніколаса де Брейна, чия система AUTOMATH (1967) стала першим механізмом комп'ютеризованого доведення теорем у фіброваній моделі. Хенк Барендрегт розвинув цю ідею, створивши лямбда-куб — класифікацію типових систем, де найвища вершина (Calculus of Constructions, CoC) уможливила вищі формальні мови. Ми використовуємо CoC як базис для теоретичного ядра, доповнюючи його вищими мовами для поглинання машинною верифікацією усієї доступної математики підприємства.

13.1 Формальна верифікація та валідація

Для унеможливлення помилок на виробництві застосовуються різні методи формальної верифікації. Формальна верифікація — доказ, або заперечення відповідності системи у відношенні до певної формальної специфікації або характеристики, із використанням формальних методів математики.

Згідно з міжнародними нормами (IEEE, ANSI)[1] та у відповідності до вимог Європейського Аерокосмічного Агенства[2], процес валідації

[1] IEEE Std 1012-2016 — V&V Software verification and validation

[2] ESA PSS-05-10 1-1 1995 – Guide to software verification and validation

включає в себе перегляд (code review), тестування (модульне, інтеграційне, властивостей), перевірку моделей, аудит — увесь комплекс необхідний для доведення, що продукт відповідає вимогам висунутим при розробці.

Огляд сучасних засобів формальної верифікації показує, що найкращі результати досягаються тоді, коли мова програмування сама слугує інструментом математичного доведення. Класифікація цих підходів виділяє окрему *спектральну категорію мов формальної верифікації* (Dependently Typed Systems). Процес розподіляється на валідацію (чи правильну систему ми будуємо?) та верифікацію (чи правильно ми її побудували?).

13.2 Формальна специфікація

Для спрощення процесу верифікації та валідації застосовується математична техніка формалізації постановки задачі — формальна специфікація. Це математична модель, створена для опису систем, визначення їх основних властивостей, та інструментарій для перевірки цих систем.

Існують два фундаментальні підходи: 1) Алгебраїчний підхід, де система описується в термінах операцій та відношень між ними (аналітичний метод); 2) Модельно-орієнтований підхід, де модель створена конструктивними побудовами на базі теорії множин, а системні операції визначаються тим, як вони змінюють стан системи (синтетичний метод).

13.2.1 Програмне забезпечення та логіка

Найбільш стандартизована та прийнята в області формальної верифікації — це нотація Z[3] (Spivey, 1992), приклад модельно-орієнтованої мови, названої на честь Ернеста Цермело.

Інша відома мова формальної специфікації як стандарт для моделювання розподілених систем, таких як телефонні мережі та протоколи, це LOTOS[4] як приклад алгебраїчного підходу. Ця мова побудована на темпоральних логіках та поведінках залежниих від спостережень. Інші мови специфікацій, які можна відзначити тут — це TLA+, CSP, CCS (Milner, 1971), Actor Model, BPMN, тощо.

13.2.2 Математичні компоненти

Перші системи комп'ютерної алгебри розроблялися ще під PDP-6: MATHLAB, MACSYMA, SCRATCHPAD, REDUCE, SACLIB, MUMATH. Сучасні системи включають AXIOM, MAGMA, MUPAD, GAP.

[3] ISO/IEC 13568:2002 — Z formal specification notation
[4] ISO 8807:1989 — LOTOS

13.3 Формальні методи верифікації

Можна виділити три підходи до верифікації: 1) спеціалізовані верифікатори моделей, або системи моделювання (VST, NuPRL, TLA+, Twelf, SystemVerilog); 2) алгебраїчні мови для синтетичних моделей і глибокого вбудовування (Coq, Agda, Lean, F*, cubicaltt, RedPRL); 3) системи автоматичного доведення теорем і синтезу програм (HOL/Isabelle, ACL2).

13.3.1 Алгебраїчні мови та System F

Вступ у теорію мов програмування неможливий без розуміння фундаменту типізації, побудованого на поліморфному лямбда-численні **System F**. Завдяки System F, програми автоматично гарантують відсутність цілих класів помилок періоду виконання. Компілятор розв'язує рівняння рівності типів через механізми уніфікації, працюючи у безпечній та математично обґрунтованій категорії.

13.4 Моделі процесів

Для моделювання складних бізнес-процесів (BPMN) та високонавантажених мережевих протоколів ми спираємося на математичну формалізацію, здатну автоматизувати генерування тестів. Для цього розроблено спеціалізовану мову специфікацій **Zen Crypted Dharma DSL (Z.180)**, яка слугує виконавчим доповненням до міжнародного стандарту **ITU-T Z.120 Message Sequence Chart (MSC)**.

Мова Z.180 абстрагує транспортний рівень та криптографічні кодування, природним чином маплячись на конструкти Z.120: сесії (Sessions) стають інстансами (lifelines), дії (Actions) — вихідними/вхідними повідомленнями, а експектації (Expectations/Predicates) — умовами перевірки. Завдяки цьому забезпечується строга верифікація міжшарових інваріантів (консистентність курсорів читання, механізми ABAC, модерації) без втручання в імутабельну історію подій. Цей підхід безпосередньо сумісний з TTCN-3 для генерації тест-сьютів і формальною семантикою процес-алгебр з Annex B. Процес генерації інтеграційних шлюзів та SDK на основі цих специфікацій стає рутинною механічною трансформацією з AST у клієнти Swift, Kotlin, Erlang, C# та JavaScript.

13.5 Формальні мови та середовища виконання

Усі середовища виконання можна умовно розділити на два класи: 1) інтерпретатори нетипізованого або просто типізованого лямбда числення (Erlang/BEAM, V8, HotSpot, Kx, PyPy); 2) безпосередня

генерація інструкцій процессора і лінкування (OCaml, Rust, Haskell, Pony).

Найбільш цікаві цільові платформи для виконання программ які побудовані на основі формальних доведень для нас є OCaml (основна мова екстракту Coq), Rust (відсутність сміттєзбірника), Erlang (неблокована семантика π-числення) та Pony.

13.5.1 Формальні інтерпретатори та екстракція

Рантайм може будуватися як інтерпретатор нетипізованої системи. У рамках розвитку проєкту використовується PTS тайпчекер Henk (на базі Erlang/OCaml) у якості проміжної мови для повної нормалізації лямбда термів. Тотальна програма виступає способом лінкування з підсистемою вводу-виводу віртуальної машини Erlang, поєднуючи функціональну довершеність (CoC) з реальною неблокованою роботою.

13.6 Базова схема підприємства ERP/1

Генерація, валідація та верифікація зливаються в єдиний обчислювальний конвеєр для бази підприємства. Та встановлено, що усі системи доведення і середовища виконання є носіями мовних елементів, які згідно теорії типів Мартіна-Льофа можна кластеризувати так:

- O_λ — нетипизоване λ-числення Чорча;
- O_π — числення процесів, CCS, CSP або π-числення Мілнера;
- O_μ — тензорне числення та векторизація;
- O_Π — числення конструкцій (функціональна повнота);
- O_Σ — числення контекстів (контекстуальна повнота);
- $O_=$ — теорія типів Мартіна-Льофа (логіка);
- O_W — числення індуктивних конструкцій (матіндукція);
- O_I — гомотопічна система типів (формальна математика).

Усі бізнес-правила (BPMN/BPE), протоколи міжсервісної взаємодії та політики доступу (ABAC) проходять етап символьного виконання, тестування за моделлю MSC та типового розв'язання. Лише код, який успішно формує доведення власної консистентності у термінах цих вищих мов програмування, допускається до середовища виконання підприємства.

Розділ 14

Інфраструктурний рівень безпеки інтернету

14.1 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].

[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.

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.

---

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

14.1.1 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/rOvysLkBHVNB9CuTISwfBO=",
     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: 54004158142501264913150880415587183761387741926 8,
     contentTimestamp: {"1.2.840.113549.1.7.2",
      369952533463044024072847521118748970 26, "20240221110626Z",
      "MfvlhoDVCPkptQRN+S2zNGp0nrOsS93mLdbcz/kZ9GI="}
   ], "АТ КБ \"ПРИВАТБАНК\"", "", "", ~c"UA", "Київ"}
]
```

14.2 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.

14.2.1 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
```

14.2.2 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
```

14.2.3 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
```

14.2.4 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
```

14.2.5 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
```

14.2.6 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).
```

14.2.7 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
```

14.3 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
```

14.3.1 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
```

14.3.2 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.

14.3.2.1 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
```

14.3.2.2 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.

14.3.2.3 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
```

14.3.2.4 P10CR/CP

Запускаємо сервер та генеруємо сертифікати CA та CSR користувача:

```
mixdeps.get 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.pem — csrclient.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
```

14.3.2.5 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'
```

14.3.2.6 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
```

14.3.2.7 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
```

14.3.2.8 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"
```

14.3.2.9 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.

14.3.3 СА, АЦСК, ЦЗО та ОЗО

Акредитований центр сертифікації ключів (АЦСК) в українській термінології та Certificate Authority (CA) в міжнародній виконують спільну задачу — управління життєвим циклом інфраструктури відкритих ключів. В Україні розбудова цієї інфраструктури спирається на Центральний засвідчувальний орган (ЦЗО), який відповідає за адміністрування кореневого сертифіката держави, та Окремі засвідчувальні органи (ОЗО), що є підлеглими сертифікаційними вузлами. ЦЗО формує довіру на національному рівні та інтегрується з між-

народними реєстрами, тоді як ОЗО забезпечують функціонування конкретних галузевих доменів.

Для забезпечення цих процесів розроблено рішення `synrc/ca` — компактний та потужний засвідчувальний центр розміром усього 2000 рядків коду (*Pure Elixir*). Він комплексно покриває потреби інфраструктури:

- **Підтримка PKI-сутностей:** CA, RA, SERVER, CLIENT, HUMAN, PROGRAM.
- **Адаптація державних стандартів:** Повна підтримка операцій у поліноміальному базисі Галуа $\mathsf{GF}(2^m)$ для криптографії ДСТУ 4145-2002.
- **Широкий спектр еліптичних кривих та схем:** `secp256k1`, `secp384r1`, `secp521r1` та підтримка гібридних механізмів Діффі-Геллмана (`RSA`, $\mathsf{GF}(p)$, $\mathsf{GF}(2^m)$).
- **Документи розширеної ідентифікації (EUID):** TAXID (РНОКПП), PID (паспорт), IBAN, HIID, LOYAL.
- **Мережева взаємодія:** Обслуговування запитів за протоколами CMP (TCP 8829), EST (HTTP 8047) та CMC (TCP 5318) згідно з RFC.

14.4 Безпечна система доменних імен DNSSEC

Безпека маршрутизації та ідентифікації ресурсів в інтернеті критично залежить від цілісності доменної системи імен. Протокол DNSSEC (Domain Name System Security Extensions) додає криптографічні підписи до записів DNS, що унеможливлює підміну IP-адрес (cache poisoning). В інфраструктурі державних інформаційних систем впровадження DNSSEC є обов'язковим для захисту урядових доменів (`gov.ua`), гарантуючи, що звернення громадян до криптографічних сервісів є автентичними та надійно захищеними від атак перехоплення (Man-In-The-Middle).

14.5 Захищений месенджер SYNRC CHAT

Рішення `synrc/chat` демонструє застосування інфраструктурних стандартів безпеки для створення захищеного корпоративного месенджера. Він базується на імплементації протоколу *Zen Crypted Buddha* та глибоко інтегрований з екосистемою Erlang/OTP (`Mnesia` для маршрутизації та зберігання повідомлень).

Головною відмінністю SYNRC CHAT є наскрізне використання сертифікації:

- Використання X.509 CMS Envelopes для управління ключами та захисту вмісту.
- Імплементація ASN.1 визначених протоколів та DER бінарної серіалізації поверх класичних транспортів.
- Підтримка сучасних криптографічних примітивів: X25519, X448, SECP384r1.
- Вбудована інтеграція з CA через протоколи CMP та EST для автоматичного отримання та оновлення сертифікатів клієнтів за допомогою еліптичної криптографії.

За замовчуванням кожен пакет у месенджері підписується (Sign/Verify) та шифрується (Encrypt/Decrypt), що перетворює його на інструмент з рівнем гарантії безпеки (*High Assurance*), придатний для систем розвідки та оперативного управління.

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

14.6 Система директорії підприємства LDAP

Пройшло 13 років з того часу як компанія SYNRC випустила була свій власний брендований LDAP Directory Server на Erlang з під-

тримкою 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. Сучасна версія `synrc/ldap` реалізована всього у 300 рядках коду на Elixir, але при цьому забезпечує абсолютну сумісність із низкою фундаментальних IETF RFC (2849, 3296, 3671-3673, 4510-4518, 5480). Архітектура підтримує гаряче перемикання бекендів зберігання даних: SQLITE, LMDB та MONGODB, задовольняючи як вимоги до вбудовуваних рішень (in-memory SQLite), так і високонавантажених кластерів (LMDB).

14.6.1 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 (атри-

бутами) 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.

14.6.2 Предметна область

14.6.2.1 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.

14.6.2.2 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 яції побудований також Windows Registry, Microsoft Access, та можливо і інші внутрішні продукти компанії Microsoft.

14.6.2.3 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..

14.6.3 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 дійсню 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_inet:async_accept, такий як 5HT/tcp, але зрештою відмовився так як не вбачаю в надмірному контролі семантики процесу ніяких перевах, для нас діє така сама семантика як в вебі, відвалився клієнт і нічого страшного, нікому його контроль на рівні сигналінгу не потрібен. Ми не слідкуємо за LDAP клієнтами!

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
```

14.6.3.1 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
```

14.6.3.2 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
```

14.6.3.3 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 виманає 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.

14.6.3.4 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
```

14.6.3.5 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
```

14.6.3.6 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
```

14.6.3.7 SEARCH

Операція `search` є фундаментальною базою для читання та вибірки даних у директорії LDAP. Вона підтримує складні фільтри, які визначаються рекурсивним типом `Filter`. У цій системі запит парситься та транслюється у внутрішнє представлення, яке у свою чергу генерує оптимізовані SQL-запити. Підтримуються всі стандартні оператори фільтрації: `present`, `equalityMatch`, `substrings`, а також логічні оператори `AND`, `OR` та `NOT`, що дозволяє дуже гнучко шукати записи у корпоративному дереві та делегувати швидкодію індексам підлеглої реляційної СКБД.

14.6.3.8 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
```

14.6.3.9 ABANDON/UNBIND

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

14.6.4 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.

14.7 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.

14.7.1 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.

14.7.2 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
```

14.7.3 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.

14.7.3.1 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.

14.7.3.2 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.

14.7.3.3 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.

14.7.4 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).

14.7.5 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.

14.7.5.1 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 віртуальниxx 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.

14.7.5.2 Послідовності та множини

Типи `SEQUENCE` та `SET` відіграють роль структур. При їх розборі компілятор генерує код, який обробляє кожен елемент об'єкта або строго по черзі (для `SEQUENCE`), або у довільному порядку (для `SET`). Враховуючи опціональність тегів, десеріалізатор повинен гнучко опрацьовувати пропущені поля, спираючись на контекстно-залежні теги.

14.7.5.3 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].

14.7.5.4 Імплісіти, експлісіти та опшинали

Якщо ви хочете позначити поле додатковою інформацією, такою як можливість приймати значення відсутності (`OPTIONAL`) та вид тегування (`IMPLICIT` або `EXPLICIT`), необхідно сформувати правильний контекст розбору. Такі теги модифікують оригінальний універсальний тег об'єкта, вимагаючи від парсера правильної маршрутизації бат-коду під час десеріалізації потоку.

14.7.5.5 Переліки та цілочисельні переліки

Типи `ENUMERATED` на макрорівні зіставляються зі стандартними алгебраїчними типами в мові програмування. Компілятор гарантує,

що значення, закодоване як ціле число, перетворюється на строгу типізацію цільової мови (наприклад, `enum` у Swift або Erlang атоми).

14.7.5.6 Тензори

Масиви даних (тензори) в моделі ASN.1 кодуються як `SEQUENCE OF` або `SET OF`. Компілятор перетворює їх у списки (List) або масиви (Array). Оскільки тип може містити довільну кількість елементів, десеріалізація вимагає циклічного читання структури до досягнення маркеру кінця інформаційного потоку.

14.7.6 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_ber парсерів на C++ і в 2 рази швидші за парсери asn1c.

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#.

14.8 Протокол розмежування доступу ABAC

Модель контестного або атрибутного доступу (Attribute-Based Access Control, ABAC) забезпечує гнучке розмежування повноважень. На відміну від класичної рольової моделі (RBAC), де доступ фіксується виключно за роллю користувача, ABAC використовує атрибути суб'єкта, об'єкта, операції та глобального середовища для динамічного прийняття рішень під час виконання запиту.

Реалізація `erpuno/abac` повністю відповідає міжнародному стандарту **ISO/IEC 29146:2016** (Information technology — Security techniques — A framework for access management) та рекомендаціям **NIST SP 800-162**. Архітектура бібліотеки структурно розподіляє відповідальність між базовими компонентами системи.

Запит на доступ формалізується структурою `request`, що містить `subject` (ідентифікатор користувача, сервісу або ERP Employee) та `endpoint` (API-метод або дія: VIEW, EDIT, SIGN тощо).

14.8.1 PEP (Policy Enforcement Point)

Точка забезпечення виконання (PEP, в рамках `ABAC.API`) — це програмний інтерфейс, який перехоплює запит користувача на доступ до захищеного ресурсу. Його головна мета — призупинити виконання бізнес-операції та викликати центр прийняття рішень для отримання дозволу. PEP діє як суворий шлюз, блокуючи дію суб'єкта за відсутності позитивного вердикту.

14.8.2 PIP (Policy Information Point)

Точка інформації про політики (PIP) — це системний інтерфейс для отримання додаткових даних (`ABAC.PIP`). Коли бракує атрибутів для перевірки політики (наприклад, вік користувача, геопозиція чи посада), ініціюється звернення до PIP. Він автоматично збагачує запит атрибутами з поточного контексту екосистеми (модулі ERP, BPMN-двигун BPE, реляційні бази даних або директорії LDAP).

14.8.3 PAP (Policy Administration Point)

Точка адміністрування (PAP) — компонент, що забезпечує створення, зберігання та аудит самих політик безпеки. У вітчизняному стеку для безвідмовного та високошвидкісного збереження правил доступу використовується масштабовна база даних ключ-значення `KVS`.

14.8.4 PDP (Policy Decision Point)

Точка прийняття рішень (PDP, в рамках `ABAC.PDP`) — математичне ядро моделі. У цей компонент завантажені всі доступні в організації політики доступу. Коли надходить запит від PEP, PDP агрегує атрибути, обчислює логічні предикати та виносить обґрунтований фінальний вердикт (Permit або Deny) щодо кожної поточної транзакції.

Розділ **15**

Профілі безпеки NIST і їх оцінка відповідності

Комплексна система захисту інформації (КСЗІ) є невід'ємним елементом побудови державних інформаційних систем України. Для отримання *Атестата відповідності* система повинна пройти чітко регламентовану процедуру, яка охоплює розробку цільового профілю безпеки, імплементацію технічних і криптографічних засобів захисту та незалежну державну експертизу. Нормативно-правовою базою цього процесу слугують нові стандарти ТЗІ, повністю гармонізовані з міжнародним каталогом **NIST SP 800-53**.

15.1 Безпекові переваги платформи Erlang

Платформа `synrc/ca` розроблена на базі віртуальної машини **BEAM (Erlang/OTP)**, створеної спеціально для безперервних телекомунікаційних систем. Це архітектурне рішення виключає цілі класи критичних вразливостей (CVE) та дозволяє забезпечити нативну підтримку ключових контролів безпеки NIST, які принципово недосяжні на віртуальних машинах CLR та JVM:

- **SI-16 — Захист пам'яті.** Відсутність ручного керування пам'яттю повністю виключає вразливості типу *Buffer Overflow* та *Use-After-Free*, які є найчастішим вектором атак у класичних C/C++ серверах.
- **SC-39 — Ізоляція процесу.** Мільйони легковагових акторів (процесів) ізольовані на рівні віртуальної машини. Збій або компрометація одного процесу ніколи не вплине на інші, забезпечуючи абсолютну стійкість. Кожен BEAM-процес має власний збирач сміття, що відрізняє цю модель від JVM та CLR.
- **SI-13 — Запобігання збоїв.** Філософія «Let it crash» та потужні дерева супервізорів (Supervision Trees) забезпечують автоматичне відновлення стану системи за мілісекунди без простоїв.

- **AC-25 — Опорний монітор.** Маршрути, правила ABAC та критичні реєстри компілюються прямо в байт-код (memory-resident). Це виключає затримки бази даних і гарантує неможливість обходу політик (non-bypassable).

- **SC-5 — Захист від відмови в обслуговуванні.** На відміну від JVM/CLR із паузами Stop-The-World, кожен BEAM-процес має мікро-GC. Витискальна багатозадачність гарантує Soft Real-time відгук і захист від Application DoS-атак.

- **SI-2 — Усунення вразливостей (Hot Code Swapping).** Дозволяє замінювати скомпільований код (патчі) «на льоту» без рестарту VM. Вразливості закриваються або логіка змінюється без розриву активних з'єднань клієнтів.

15.2 Гомотопічне кодування ASN.1 DER

Стандарт **ASN.1 DER** обраний як фундаментальний формат серіалізації для всіх API-ендпойнтів екосистеми `synrc/ca`. На відміну від Protobuf / gRPC та JSON, які є вразливими до маніпуляцій через множинність варіантів кодування одного й того ж об'єкта, гомотопічний (канонічний) **DER** математично гарантує *єдине унікальне байтове представлення* для кожної структури даних. Це повністю виключає атаки типу *Parser Differentials* (наприклад, Bleichenbacher's forgery) та забезпечує виконання наступних вимог безпеки:

- **SI-10 — Перевірка вводу інформації.** Сувора перевірка вхідних даних відкидає будь-які нестандартні (BER) або надлишкові кодування, унеможливлюючи вразливості до маніпуляцій (malleability).

- **SI-7 — Цілісність інформації.** Цифрові підписи та хешування залишаються валідними лише за умови бієктивної (гомотопічної) серіалізації, захищаючи від прихованої модифікації байтового представлення.

- **SC-13 — Криптографічний захист.** Вимоги стандартів (X.509, CMS, PKCS) для безпечного застосування криптографії та збереження довіри до інфраструктури відкритих ключів (PKI).

15.3 Профілі безпеки КСЗІ

Система `synrc/ca` підтримує автоматичну побудову профілів Комплексної системи захисту інформації (КСЗІ) на основі нормативно-правових актів у галузі технічного захисту інформації (ТЗІ НПА).

Це дозволяє автоматизувати створення документації та мапінг вимог законодавства на технічні налаштування. Усі сертифікати видаються з імпринтами OID відповідних профілів безпеки.

15.3.1 Імплементація опорного монітора (AC-25)

Для гарантування цілісності та неможливості обходу (non-bypassable), динамічні політики безпеки (ABAC) не зберігаються у традиційних базах даних. Натомість в ERP/1 Сертифікати матриця доступу та політики компілюються безпосередньо у незмінний (immutable) BEAM байт-код віртуальної машини Erlang, функціонуючи як memory-resident механізм. Це архітектурне рішення забезпечує виконання вимог NIST щодо захисту політик від модифікації (tamper-proof) та повністю нівелює класичні вектори атак на корупцію пам'яті, притаманні традиційним ANSI C99 реалізаціям, усуваючи відповідні класи вразливостей (CVE).

15.3.2 Стандарти NIST та ISO

Застосування контролів базується на наступних міжнародних документах:

- **NIST SP 800-53** — Контроли безпеки та приватності для федеральних інформаційних систем і організацій.
- **NIST SP 800-53A** — Оцінювання засобів контролю безпеки в організаціях.
- **NIST SP 800-37** — Структура управління ризиками (Risk Management Framework, RMF).
- **FIPS 199 / FIPS 200** — Категоризація та мінімальні вимоги до безпеки федеральних систем.
- **NIST SP 800-162** — Настанови щодо систем управління доступом на основі атрибутів (ABAC).
- **ISO/IEC 27001** — Системи управління інформаційною безпекою — вимоги.
- **ISO/IEC 27002** — Інформаційні технології — методи захисту — контролі інформаційної безпеки.

Параметри безпеки включають: автоматизовану прив'язку політик безпеки до метрик; динамічний контроль доступу на основі атрибутів; неперервний аудит, логування подій та моніторинг; безперервну верифікацію та валідацію криптографічних засобів і сертифікатів.

15.4 Походження архітектури контролів: NIST vs ISO

Класифікація сімейств заходів захисту, яка лежить в основі платформи (абревіатури AC, AU, SC, IA тощо), *не є частиною* міжнародних стандартів ISO/IEC 27001 або 27002. Ці стандарти мають інший підхід до групування. Архітектура `synrc/ca` базується на еталонному каталозі **NIST SP 800-53** (США). Саме ця гранулярна база контролів була нещодавно повністю адаптована Державною службою спеціального зв'язку та захисту інформації України у новому національному стандарті **НД ТЗІ 3.6-006-24**. Шлях NIST обраний тому, що він забезпечує максимальну деталізацію, необхідну для побудови military-grade систем та атестації державних реєстрів.

15.4.1 Державний сектор (НД ТЗІ 3.6-006-24)

У державному секторі України захист інформації регламентується стандартом НД ТЗІ 3.6-006-24, повністю гармонізованим з NIST SP 800-53. Процедура побудови КСЗІ передбачає:

1. Розробку та затвердження **Цільового профілю безпеки** системи на основі Базового або Галузевого профілю.
2. Впровадження засобів технічного і криптографічного захисту, що відповідають обраним заходам контролю.
3. Проведення державної експертизи КСЗІ (або декларування відповідності) для отримання **Атестата відповідності** від Держспецзв'язку.

Нормативна документація: НД ТЗІ 3.6-006-24, НД ТЗІ 2.3-025-24 (Том 1, Том 2, Том 3).

15.4.2 Приватний сектор (NIST SP 800-53)

Для корпоративного сектору, фінтеху та критичної інфраструктури еталоном є пряме застосування сімейства стандартів NIST. Сертифікація базується на Risk Management Framework (NIST SP 800-37) і включає:

1. Категоризацію системи за FIPS 199/200 та вибір базового профілю контролів (Baseline) за NIST SP 800-53.
2. Розробку System Security Plan (SSP) та імплементацію архітектури безпеки і засобів моніторингу.
3. Незалежну оцінку контролів за NIST SP 800-53A та формальне надання внутрішнього дозволу на експлуатацію (**Authority to Operate — ATO**).

15.5 Процес управління ризиками (Risk Management Framework)

Життєвий цикл побудови та підтримки комплексної системи безпеки за методологією NIST (зокрема згідно з **NIST SP 800-37**) визначається структурою управління ризиками (Risk Management Framework, RMF). Цей безперервний процес гарантує, що заходи безпеки відповідають архітектурі, бізнес-вимогам та поточним загрозам, і складається з 6 базових кроків:

1. **Категоризація інформаційної системи (Categorize Information System).** На цьому етапі визначається рівень критичності системи (Low, Moderate, High) на основі аналізу наслідків від потенційної втрати конфіденційності, цілісності чи доступності інформації (відповідно до FIPS 199 та FIPS 200).

2. **Вибір заходів безпеки (Select Security Controls).** На основі визначеної категорії обирається базовий набір контролів безпеки (Baseline) з каталогу NIST SP 800-53, який потім адаптується (tailoring) під специфічні умови експлуатації та ризики організації.

3. **Впровадження заходів безпеки (Implement Security Controls).** Технічна та організаційна імплементація обраних контролів у системі. Розробляється детальний System Security Plan (SSP), що описує, як саме реалізовано кожен конкретний захід.

4. **Оцінювання заходів безпеки (Assess Security Controls).** Незалежна оцінка та аудит впроваджених контролів за допомогою методології NIST SP 800-53A. Визначається, наскільки коректно імплементовані захисні механізми та чи досягають вони бажаного результату.

5. **Авторизація інформаційної системи (Authorize Information System).** Офіційне прийняття залишкових ризиків керівництвом організації (Authorizing Official) на основі результатів оцінювання та надання формального дозволу на експлуатацію системи (Authority to Operate, ATO).

6. **Моніторинг заходів безпеки (Monitor Security Controls).** Безперервне відстеження ефективності контролів, фіксація змін у системі та її середовищі, аналіз інцидентів безпеки та періодичне оновлення SSP. Цей крок гарантує, що система зберігає відповідність вимогам протягом усього свого життєвого циклу.

15.6 Принцип розділення обов'язків

Відповідно до нормативної бази України (НД ТЗІ 2.6-001-11) та найкращих світових практик (принцип *Maker/Checker*), побудова та сертифікація КСЗІ є строго **двоетапною процедурою**. Вона вимагає залучення двох абсолютно незалежних провайдерів-ліцензіатів для унеможливлення конфлікту інтересів.

15.6.1 Етап 1 — Розробка профілю безпеки

Провайдер №1 проводить оцінку ризиків, обстеження середовища та створює технічні інструкції — **Технічне завдання (Цільовий профіль безпеки)**. Здійснює фізичну імплементацію архітектури безпеки: налаштовує криптографію, Reference Monitor (AC-25), встановлює сертифікати та здає систему в дослідну експлуатацію.

15.6.2 Етап 2 — Атестація відповідності

Провайдер №2 отримує Технічне завдання від розробника та розробляє **Програму експертизи**. Проводить незалежний технічний аудит (атестація відповідності), інструментальне тестування та перевірку імплементованих контролів наживо. У разі успіху видає Експертний висновок для Держспецзв'язку.

15.7 Каталог профілів безпеки ERP/1

Нижче наведено перелік розроблених профілів безпеки, кожен з яких визначає цільовий або галузевий набір контрольних елементів на базі НД ТЗІ 2.3-025-24 та НД ТЗІ 3.6-006-24:

- **FULL DIRECTORY (1123 контроли)** — повний каталог контрольних елементів із трьохтомника НД ТЗІ 2.3-025-24. Еталонний документ для побудови будь-якого галузевого або цільового профілю.
- **OPEN & CONFIDENTIAL — 126 контролів** — базовий профіль безпеки відкритої та конфіденційної інформації, затверджений Наказом Адміністрації Держспецзв'язку №409 від 30.06.2025.
- **OFFICIAL — 155 контролів** — базовий профіль безпеки службової інформації, затверджений Наказом Адміністрації Держспецзв'язку №419 від 02.07.2025.
- **COURT INDUSTRY PROFILE (190 контролів)** — галузевий профіль для судової системи. Описує розширений набір базових контролів для обробки чутливої інформації в державних реєстрах та судах, враховуючи вимоги високої доступності та цілісності даних на базі НД ТЗІ 3.6-006-24.

- **DIGITAL UKRAINIAN GOVERNMENT INDUSTRY PROFILE (350 контролів)** — галузевий профіль безпеки Міністерства цифрової трансформації України. Описує розширений набір базових контролів для обробки чутливої інформації в державних реєстрах та судах на базі НД ТЗІ 3.6-006-24.

- **ERP/1 SECURITY TARGET** — цільовий профіль для телекомунікаційної системи ERP/1. Описує імплементацію управління доступом на основі атрибутів (ABAC), управління інформаційними потоками (BPE), інфраструктури відкритих ключів (PKI) та надійного аудиту (KVS) для безпеки критичних бізнес-процесів організації.

- **X.509 CMS MESSAGING SECURITY TARGET** — цільовий профіль для систем корпоративного месенджера. Базується на криптографічних протоколах X.509 та CMS (Cryptographic Message Syntax), забезпечуючи наскрізне шифрування (E2EE), строгу автентифікацію та гарантію невідмовності авторства для юридично значущих комунікацій.

- **VPN SECURITY TARGET** — цільовий профіль для інфраструктури віртуальних приватних мереж (VPN) та центрів сертифікації (PKI). Визначає механізми тунелювання, автентифікації вузлів (SC-8) і захисту мережевого трафіку на транспортному рівні для захисту каналів зв'язку від перехоплення та модифікації.

15.8 Архітектура EUDI та децентралізована PKI

EUDI — це децентралізований PKIX з контролем атрибутів на рівні ABAC, що використовує JSON для кодування та HTTP як транспорт. Система визначає наступних учасників:

- **eIDAS Node** — державний центр сертифікації (CA);
- **EUDI Verifier** — перевірка Verifiable Presentations;
- **EUDI Wallet (Holder)** — iOS/Android застосунок для зберігання та презентації облікових даних;
- **EUDI Provider (Issuer)** — OpenID для Verifiable Credentials (PID, QEAA);
- **Personal Identification Data (PID) Provider** — державне підприємство «Дія»;
- **QEAA / EAA** — кваліфіковані та некваліфіковані атестати атрибутів;
- **Qualified Electronic Signature Provider (QP)** — кваліфіковані сертифікати (QC).

15.8.1 Holder, Issuer, Verifier

В екосистемі OpenID4VC Verifier та Issuer зв'язані опосередковано через життєвий цикл облікових даних, а взаємодія між ними в основному здійснюється через Holder. Ця архітектура забезпечує довіру без прямих, постійних відносин між Verifier та Issuer, дотримуючись принципів конфіденційності та децентралізації.

Верифікатор не контактує безпосередньо з видавцем під час типових операцій, якщо тільки не потрібна перевірка статусу. Holder виступає посередником, забезпечуючи конфіденційність та контроль над даними, що передаються. EUDI Wallet виступає як Holder, а QEAA, EAA, PIP (TSPs) — як EUDI Providers або Issuers. EUDI Verifier виконує перевірку статусу облікових даних та верифікацію презентацій.

15.8.2 PKIX vs OpenID4VC

Модель EUDI має схожість із PKIX: так само, як особа використовує підписаний набір атрибутів (сертифікат X.509 зі CSR-атрибутами) для автентифікації та авторизації в PKI, провайдер OpenID4VC (PIP) охоплює набір атрибутів (цифрова презентація claims) та видає електронні документи у форматі mDOC для EUDI Wallet.

Однак, на відміну від PKIX з його централізованою моделлю, EUDI забезпечує розподілену модель без єдиного кореневого CA, де всі сторони пов'язані криптографічно. EUDI також здійснює більш

тонкий та строгий контроль над атрибутами (claims), подібно до моделі ABAC.

CRL та OCSP можуть створювати проблеми конфіденційності, оскільки передбачають запит до CA, що потенційно розкриває активність користувача. OpenID4VC пом'якшує це, дозволяючи Holder керувати процесом, а деякі реалізації уникають перевірки статусу в режимі реального часу, включаючи криптографічні докази безпосередньо в облікові дані.

15.9 Відповідність вимогам GDPR та захист персональних даних

При інтеграції державних ІКС України з європейським цифровим простором обов'язковим є дотримання вимог регламенту GDPR (General Data Protection Regulation). Платформа ERP/1 реалізує наступні принципи GDPR:

- **Privacy by Design та Privacy by Default**: Обробка персональних даних мінімізована на рівні протоколів. Усі транзакційні повідомлення в Cartulary (Картулярії) деперсоналізовані та використовують хеш-ідентифікатори користувачів.

- **Обмеження цілей та мінімізація даних**: Атрибути користувача в каталогах LDAP зберігаються за моделлю ABAC. Сервіси отримують доступ лише до тих полів даних, які необхідні для виконання конкретної бізнес-функції (наприклад, перевірки віку без передачі дати народження за технологією Zero-Knowledge Proofs).

- **Право на забуття (Right to be Forgotten)**: В архітектурі незмінних реєстрів (на базі блокчейну або захищеного Картулярію) фізичне видалення записів є неможливим через вимоги цілісності ланцюжка. ERP/1 вирішує цю суперечність за допомогою *криптографічного стирання* (Cryptographic Erasure) — персональні дані шифруються унікальним симетричним ключем суб'єкта, і при реалізації права на забуття цей ключ безповоротно знищується з IAS, що робить розшифрування персональних даних неможливим.

15.10 Тестові питання та завдання

1. Які особливості віртуальної машини BEAM (Erlang) забезпечують нативне виконання вимог NIST SP 800-53 щодо захисту пам'яті (SI-16) та ізоляції процесів (SC-39)?

2. Чому гомотопічне (канонічне) кодування DER є більш безпечним для цифрових підписів порівняно з гнучким кодуванням BER чи JSON?

3. Опишіть ролі Holder, Issuer та Verifier в архітектурі EUDI Wallet та порівняйте їх із класичною моделлю сертифікатів X.509.

4. Як реалізується вимога GDPR щодо «права на забуття» в системах з незмінними реєстрами (immutable ledgers)?

5. **Практичне завдання**: Опишіть перелік OID атрибутів для сертифіката користувача, які необхідні для авторизації за моделлю ABAC у системі доступу до медичних карток.

Розділ **16**

Апробація та промислове розгортання

16.1 Історія впроваджень в органах сектору безпеки

Компоненти та архітектурні рішення платформи «ERP/1» успішно пройшли етапи проектування, розгортання та тривалої промислової апробації у державних інформаційних системах органів сектору безпеки та оборони України:

- **ЄРЗ** (Єдиний реєстр зброї Національної поліції України) — ведення дозвільної системи, облік цивільної зброї, інтеграція з кабінетами громадян.
- **СУСЗЦЗ** (Система управління силами та засобами цивільного захисту Державної служби України з надзвичайних ситуацій) — координація чергових змін, оперативне реагування, моніторинг ресурсів.
- **ЄІС МВС** (Єдина інформаційна система Міністерства внутрішніх справ) — інтеграційна шина та обмін даними між відомчими реєстрами.
- **ФП МТРЗ** (Функціональна підсистема матеріально-технічного та ресурсного забезпечення МВС) — автоматизація логістики, обліку ТМЦ та державного замовлення.
- **ГСЦ МВС** (Головний сервісний центр) — оптимізація черг, облік посвідчень водія та транспортних засобів.

16.2 Оркестрація Erlang-кластерів у Kubernetes

Для промислового розгортання та автоматичного масштабування (autoscaling) нод бекенду в хмарних та локальних середовищах застосовується контейнеризація на базі Docker та оркестрація в Kubernetes.

16.2.1 Динамічний пошук нод (Peer Discovery)

Оскільки Erlang-вузли вимагають постійної зв'язності для побудови розподіленого кластера Mnesia, використовується бібліотека `libcluster` з DNS-пошуком через headless сервіси Kubernetes:

```
config :libcluster,
  topologies: [
    k8s_erp: [
      strategy: Cluster.Strategy.Kubernetes.DNS,
      config: [
        service: "erp-backend-headless",
        application_name: "erp"
      ]
    ]
  ]
```

16.2.2 Порти та дистрибуція через TLS

Для безпеки розподілена взаємодія нод Erlang здійснюється через шифрований TLS-канал. Для цього налаштовується генерація сертифікатів для кожної ноди при старті контейнера та прокидаються наступні порти:

- **4369** — порт служби `epmd` (Erlang Port Mapper Daemon);
- **9100–9115** — діапазон портів для міжнодової дистрибуції.

16.2.3 Helm та Service Mesh (Istio)

Для спрощення розгортання розроблені Helm-діаграми (Helm Charts), які містять описи StatefulSet для баз даних (Mnesia) та Deployment для безстатусових серверів сесій. Istio Service Mesh інтегрується для тонкого налаштування трафіку (Traffic Splitting), моніторингу затримок (Telemetry) та взаємної mTLS авторизації між клієнтськими сервісами.

16.3 Політики інфраструктурного розгортання

Для забезпечення високої доступності, передбачуваності та автоматизації оновлень платформи «ERP/1» впроваджено суворі політики розгортання на базі концепції GitOps та підходу «Інфраструктура як код» (IaC). Технологічний стек спроектований за принципом мінімальної кількості рухомих частин та максимальної автономності.

16.3.1 Політика віртуалізації (Proxmox VE)

Усі обчислювальні ресурси платформи розгортаються на базі гіпервізора Proxmox VE з дотриманням таких вимог:

- **Мінімалістичний оверхед**: Віртуальні машини створюються виключно за технологією Thin Provisioning (LVM-Thin або ZFS) для економного споживання дискового простору та підтримки швидких знімків.
- **Декларативність та шаблони**: Впроваджено використання незмінних шаблонів ОС Ubuntu LTS з підтримкою Cloud-Init для автоматичного налаштування мережі, SSH-ключів та системних параметрів.
- **Резервування та мережі**: Ноди керування та ноди даних розподіляються по різних фізичних серверах Proxmox (антиафініті політики). Мережева топологія ізолюється за допомогою віртуальних мостів (Linux Bridges) та VLAN.

16.3.2 Політика оркестрації кластера (Kubespray)

Створення, конфігурування та оновлення версій Kubernetes-кластерів виконується за допомогою Kubespray:

- **Мінімальний набір плагінів**: Виключаються сторонні хмарні інтеграції. Використовуються стандартні компоненти: `containerd` як контейнерне середовище виконання, інтеграція з `systemd` та локальні дискові сховища.
- **Контроль версій**: Конфігураційні інвентарі Ansible (hosts.yaml) та змінні кластера (group_vars) зберігаються у центральному Git-репозиторії інфраструктури.
- **Мережеві політики**: Використовується Calico CNI з активованим NetworkPolicies для ізоляції мікросервісів у межах просторів імен Kubernetes.

16.3.3 Політика доставки додатків (ArgoCD)

Впроваджено модель Pull-based GitOps для управління станом додатків:

- **Єдине джерело істини**: Жодна зміна конфігурації чи коду не вноситься в кластер вручну через `kubectl`. Усі ресурси Kubernetes декларуються в Git.
- **Дрейф конфігурацій (Configuration Drift)**: ArgoCD здійснює безперервний моніторинг стану кластера. У разі виявлення розбіжностей між Git та кластером автоматично ініціюється процес синхронізації (Self-Healing) або надсилається сповіщення.
- **Декларативне керування додатками**: Використовується структура ресурсів типу `Application` для опису зв'язку між Git-репозиторієм та цільовим простором імен.

16.3.4 Політика керування Helm-діаграмами

Усі компоненти платформи пакуються у Helm-діаграми та зберігаються у приватному репозиторії:

- **Календарне версіонування (за принципом N2O)**: Замість класичного SemVer використовується формат версій `X.Y.Z`, де `X` — кількість «нових років», які зустрів продукт з моменту створення, `Y` — номер місяця релізу, а `Z` — день релізу або порядковий номер збірки в межах цього місяця. Використання тегів `latest` заборонено.
- **Локальне сховище**: Helm-діаграми зберігаються у власному захищеному HTTP/OCI-репозиторії, що гарантує автономність доставки у закритих контурах без доступу до глобальної мережі.

16.3.5 Політика керування DNS (synrc/ns)

Навігація та резолвінг імен всередині інфраструктури виконується за допомогою власного легковагого DNS-сервера `synrc/ns`:

- **Автономність**: DNS-сервер написаний на Erlang/Elixir, інтегрується безпосередньо в мережевий контур та працює з мінімальним використанням RAM (до 20 МБ).
- **Декларативні зони**: Усі записи доменних зон зберігаються у вигляді JSON-файлів у Git-репозиторії та автоматично оновлюються при зміні IP-адрес чи додаванні нових сервісів.

16.3.6 Політика базових образів контейнерів (Alpine Linux)

З метою мінімізації вектора атак, економії дискового простору та підвищення швидкості масштабування (cold start) нод у Kubernetes, для побудови Docker-контейнерів BEAM-додатків діє політика використання легковагого базового образу **Alpine Linux** (версії `alpine:3.20`):

- **Мінімальний обсяг образів**: Використання Alpine Linux замість повноважних дистрибутивів на кшталт Ubuntu Server дозволяє зменшити розмір базового шару образу з ~80 МБ до ~5–8 МБ.
- **Безпека**: Базовий образ містить мінімальну кількість утиліт та бібліотек, що знижує вразливість системи перед CVE-загрозами. Заборонено встановлення компіляторів чи пакетних менеджерів (наприклад, `apk`) у фінальних продуктових образах.

- **Статична лінковка**: Усі NIF-бібліотеки на C/Rust, які завантажуються Erlang-додатками, компилюються під бібліотеку `musl` C, вбудовану в Alpine.

16.4 Покомпонентне розгортання ІКС та Служб ERP/1 через Helm

Для забезпечення модульності та гнучкості керування інфраструктурою, кожен прикладний продукт екосистеми **ІКС ERP/1** та кожна складова частина **Служб безпечного зв'язку (SLUB) ERP/1** поставляються як окремі незалежні Helm-діаграми. Це дозволяє розгортати лише необхідні модулі залежно від призначення конкретного контуру (наприклад, окремо освітній контур чи медичний сервіс).

16.4.1 Пакети прикладних продуктів ІКС ERP/1

Усі 10 прикладних продуктів ІКС мають власні Helm-діаграми:

- `lms-education` (Освіта) — автоматизоване розгортання LMS-сервісу, сервісів планування занять та інтерфейсів інтеграції з ЄДЕБО.
- `hl7-health` (Здоров'я) — MIC-сервіси, що підтримують HL7 FHIR API та адаптери синхронізації з eHealth.
- `crm-documents` (Документи) — служба електронного документообігу та сховищ файлів, що тісно взаємодіє з рушієм BPE.
- `acc-accounting` (Облік) — сервіси бухгалтерії, розрахунку зарплат та кадрового обліку з підтримкою Mnesia.
- `wms-warehouse` (Склад) — служба адресного зберігання ТМЦ та логістичних операцій.
- `cart-registers` (Реєстри) — low-code рушій реєстрів, що містить інтерфейси підключення до Трембіти.
- `ai-generation` (Істотність) — сервіс семантичного аналізу та інференсу локальних LLM з підтримкою прокидання GPU (NVIDIA CUDA).
- `olap-analytics` (Аналітика) — аналітичне DWH-сховище на базі DuckDB/MonetDB.
- `pm-projects` (Проєкти) — локальна система управління задачами (Jira-alternative) та Wiki-сервер.
- `itsm-incidents` (Service Desk) — диспетчер інцидентів та SLA-контролер.

16.4.2 Пакети інфраструктурних служб ERP/1 (SLUB)

Служби комунікацій та безпеки (SLUB) забезпечують роботу транспортного та криптографічного контуру платформи:

- `ns-dns` — легковагий локальний DNS-сервер для автономного резолвінгу імен.
- `ca-pki` — локальний центр сертифікації для випуску ключів X.509.
- `vpn-wireguard` — VPN-шлюз для захисту міжнодових каналів зв'язку.
- `ldap-directory` — ієрархічний реєстр користувачів та організацій з підтримкою ABAC.
- `ias-auth` — служба ідентифікації, автентифікації та перевірки статусів сертифікатів (OCSP).
- `chat-messenger` — високопродуктивний WebSocket брокер реального часу.
- `mail-delivery` — поштовий сервер для асинхронного транспорту офіційної кореспонденції.
- `rest-bpe` — REST API шлюз для керування бізнес-процесами на базі рушія BPE.
- `abac-clearance` — Go-сервер для авторизації, перевірки допусків та збору аудит-логів доступу.
- `mach-ivr` — сценарний рушій станів телефонії та автоінформатора (IVR).
- `bpe-engine` — BPMN-рушій бізнес-процесів платформи.
- `kvs-database` — вбудоване сховище даних на базі СУБД KVS.
- `nitro-portal` — веб-портал рендерингу інтерфейсів.
- `n2o-server` — WebSocket-сервер додатків.
- `faiss-search` — векторний пошуковий рушій для інтелектуальних сервісів штучного інтелекту.
- `prometheus` — збір та збереження часових рядів метрик додатків.
- `grafana` — візуалізація метрик та побудова оперативних дашбордів моніторингу.
- `loki` — збір та горизонтальне зберігання журналів логування (logs) BEAM-вузлів.

- `otel-collector` — OpenTelemetry шлюз збору уніфікованих метрик, логів та трасувань.

16.4.3 Специфікація інфраструктури та сервісів

Для координації мікросервісів приведемо повну специфікацію портів, протоколів та типу збереження стану (statefulness) компонентів платформи (табл. 16.1).

Компонент	Namespace	Призначення	Порт	Стан
Інфраструктурні служби				
`ns-nameserver`	`core`	DNS-сервер дозволу імен	53:8101	Є
`vpn-wireguard`	`core`	VPN-шлюз міжнодового зв'язку	51820:8102	Є
`ca-authority`	`security`	Локальний PKI та випуск ключів	8201	Є
`ldap-directory`	`security`	ABAC-директорія користувачів	389:8202	Є
`abac-clearance`	`security`	Мандати та аудит логів доступу	8203	Є
`ias-authorization`	`security`	Автентифікація користувачів	8204	Є
`kvs-database`	`database`	Розподілене сховище KVS	8301	Є
`bpe-workflow`	`database`	Рушій процесів BPMN	8302	Є
`rest-openapi`	`web`	BPE REST API шлюз	8303	Нема
`mach-ivr`	`web`	Телефонія та автоінформатор (IVR)	8509	Нема
`nitro-portal`	`web`	Веб-інтерфейс рендерингу (nitro)	8510	Нема
`n2o-server`	`web`	WebSocket-сервер додатків (n2o)	8511	Нема
`prometheus`	`telemetry`	Моніторинг та збір метрик	8401	Є
`grafana`	`telemetry`	Візуалізація та дашборди метрик	8402	Нема
`loki`	`telemetry`	Збір та зберігання логів BEAM	8403	Є
`open-telemetry`	`telemetry`	OpenTelemetry шлюз збору OTLP	8404:8405:8406	Нема
Прикладні інформаційно-комунікаційні системи (ІКС)				
`cart-registers`	`erp`	Low-code рушій державних реєстрів	8501	Є
`hl7-health`	`erp`	MIC-сервіси та FHIR API (eHealth)	8502	Є
`crm-documents`	`erp`	Електронний документообіг	8503	Є
`acc-accounting`	`erp`	Бухгалтерський та кадровий облік	8504	Є
`wms-warehouse`	`erp`	Склад адресного зберігання ТМЦ	8505	Є
`lms-education`	`erp`	LMS-сервіси та інтеграція з ЄДЕБО	8506	Є
`chat-messenger`	`erp`	WebSocket брокер реального часу	8507	Є
`mail-delivery`	`erp`	Поштовий сервер транспорту	8508	Є
`olap-analytics`	`erp`	DuckDB/MonetDB сховище	8512	Є
`itsm-incidents`	`erp`	Диспетчер інцидентів та SLA	8513	Є
`pm-projects`	`erp`	Управління задачами та листами	8514	Є
Сервіси штучного інтелекту (AI & GPU)				
`ai-generation`	`ai`	Інференс LLM	8601	Нема
`faiss-search`	`ai`	Векторний пошук та ШІ-сервіси	8602	Нема

Таблиця 16.1: Специфікація інфраструктури та типів розгортання компонентів

16.4.4 Аналіз мета-схеми Kubernetes та мінімальний набір об'єктів

Для забезпечення життєвого циклу платформи в Kubernetes використовується строго обмежений мінімальний набір об'єктів мета-схеми API. Цей мінімум є оптимальним для ізольованих контурів і складається з:

- **Namespace** — ізоляція ресурсів платформи.
- **ConfigMap та Secret** — декларативне конфігурування та збереження ключів/сертифікатів.
- **Service** — логічна маршрутизація трафіку та внутрішній DNS-резолвінг.
- **Ingress** — зовнішній доступ до вебинтерфейсів через Ingress-контролер.
- **Deployment** — розгортання Stateless-мікросервісів.
- **StatefulSet (STS)** — управління Stateful-нодами СУБД та систем моніторингу.
- **PersistentVolumeClaim (PVC)** — динамічне виділення дискового простору.
- **HorizontalPodAutoscaler (HPA)** — динамічне масштабування подів під навантаженням.

16.4.4.1 Використання StatefulSet (STS) для Mnesia/KVS

Для розподіленої СУБД Mnesia та KVS вкрай важливо зберігати стабільний мережевий ідентифікатор хоста при перезапуску та стабільний зв'язок з його дисковим сховищем. StatefulSet гарантує унікальні імена нод (`kvs-database-0`, `kvs-database-1`) та їх послідовний запуск і зупинку, що запобігає виникненню аварійних станів Split-Brain.

16.4.4.2 Конфігурація HorizontalPodAutoscaler (HPA)

Для амортизації різких стрибків користувацької активності на рівні вебпорталів застосовується HPA. Нижче наведено мінімальний приклад налаштування авто-масштабування для сервісу `crm-documents` на базі використання CPU:

```
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: crm-documents-hpa
  namespace: erp
spec:
```

```
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: crm-documents
  minReplicas: 2
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 75
```

16.4.5 Ієрархія конфігурацій (values.yaml)

Для спрощення адміністрування використовується парадигма Umbrella-chart, який об'єднує всі підкомпоненти через залежності. Приклад налаштування файлу `values.yaml`:

```
global:
  domain: erp.uno
  environment: production
  tls:
    enabled: true
    secretName: erp-tls-cert

crm-documents:
  enabled: true
  replicaCount: 2
  resources:
    limits:
      cpu: 1000m
      memory: 2Gi
  bpe:
    engine: enabled

ca-pki:
  enabled: true
  storage:
    size: 10Gi
    storageClass: local-path
```

16.4.6 Відповідність репозиторіїв GitHub, ArgoCD та Helm

Для забезпечення прозорості автоматизованої доставки GitOps-процесом ArgoCD, нижче наведено таблицю відповідності вихідних GitHub-репозиторіїв коду додатків та назв відповідних ArgoCD-додатків / Helm-компонентів:

GitHub репозиторій	Додаток ArgoCD / Helm-компонент
synrc/ns	ns-dns
synrc/ca	ca-pki
zencrypted/vpn	vpn-wireguard
synrc/ldap	ldap-directory
zencrypted/ias	ias-auth
synrc/chat	chat-messenger
erpuno/mail	mail-delivery
synrc/rest	rest-bpe
zencrypted/clearance	abac-clearance
synrc/mach	mach-ivr
synrc/bpe	bpe-engine
synrc/kvs	kvs-database
synrc/nitro	nitro-portal
synrc/n2o	n2o-server
synrc/faiss	faiss-search
prom/prometheus	prometheus
grafana/grafana	grafana
grafana/loki	loki
otel/otel-collector	otel-collector
erpuno/edu	lms-education
erpuno/health	hl7-health
erpuno/crm	crm-documents
erpuno/acc	acc-accounting
erpuno/warehouse	wms-warehouse
erpuno/cart	cart-registers
erpuno/ai	ai-generation
erpuno/olap	olap-analytics
erpuno/pm	pm-projects
erpuno/itsm	itsm-incidents

Таблиця 16.2 Таблиця відповідності репозиторіїв та компонентів GitOps

16.5 Модель асинхронного ETL-процесингу черг на базі KVS

У високонавантажених державних системах транзакційна обробка реального часу (OLTP) відокремлена від аналітичної обробки та побудови звітів (OLAP). Для цього в архітектурі ERP/1 реалізовано модель асинхронного ETL (Extract, Transform, Load) процесингу транзакційних логів на базі черг вбудованого сховища KVS.

16.5.1 Компоненти черги: Воркери, Курсори та Контексти

Процес асинхронної обробки базується на трьох сутностях:

- **Асинхронний воркер** — ізольований BEAM-процес, що виконує циклічну обробку повідомлень у черзі.
- **Курсор (Cursor)** — запис у сховищі KVS, який фіксує ідентифікатор останньої успішно обробленої транзакції для конкретного воркера. Це забезпечує семантику доставки «щонайменше один раз» (At-least-once).
- **Контекст (Context)** — структура даних воркера, що зберігає параметри з'єднання з цільовою БД, сесійні ключі, ліміти пакетів (batch size) та поточний стан.

16.5.2 Специфікація структур даних (Erlang)

Модель черг та курсорів описується такими Erlang-записами:

```
-record(kvs_cursor, {
    id,
    last_read_id,
    updated_at
}).

-record(etl_context, {
    worker_id,
    stream_id,
    cursor,
    batch_size = 100,
    target_db,
    transform_fun
}).
```

16.5.3 Алгоритм ETL-циклу воркера

Асинхронний вокер працює за таким циклічним алгоритмом:

1. **Extract (Вилучення)** — воркер зчитує чергову порцію записів з KVS-стріму, починаючи з позиції `last_read_id`, збереженої в курсорі.
2. **Transform (Трансформація)** — отримані сирі бінарні дані розкодовуються з ASN.1 DER, перевіряються на цілісність (підпис КЕП) та приводяться до реляційного або стовпчикового вигляду.
3. **Load (Завантаження)** — трансформований пакет записується в аналітичне OLAP-сховище (DuckDB) через швидкі NIF-порти. Після успішного запису курсор воркера оновлюється в KVS в межах однієї транзакції.

Приклад мінімальної реалізації ETL-циклу на Erlang:

```
-module(etl_worker).
-export([process_loop/1]).

process_loop(Context) ->
    Cursor = Context#etl_context.cursor,
    LastId = Cursor#kvs_cursor.last_read_id,
    StreamId = Context#etl_context.stream_id,
    Limit = Context#etl_context.batch_size,
    case kvs:get_range(StreamId, LastId, Limit) of
        [] ->
            timer:sleep(1000),
            process_loop(Context);
        Records ->
            Transformed = [ (Context#etl_context.transform_fun)(R) || R <-
     Records ],
            case load_to_olap(Context#etl_context.target_db, Transformed) of
                ok ->
                    LastRecord = lists:last(Records),
                    NewLastId = element(2, LastRecord),
                    NewCursor = Cursor#kvs_cursor{
                        last_read_id = NewLastId,
                        updated_at = erlang:system_time(second)
                    },
                    kvs:put(NewCursor),
                    process_loop(Context#etl_context{cursor = NewCursor});
                {error, Reason} ->
                    logger:error("ETL Load failed: ~p", [Reason]),
                    timer:sleep(5000),
                    process_loop(Context)
            end
    end.
```

16.6 Практичні кейси та аналіз відмов (Post-Mortem)

16.6.1 Кейс 1: Відбиття масованої DDoS-атаки на веб-портал реєстрів

Опис інциденту: Під час запуску оновленого кабінету власника зброї система зазнала HTTP-флуду обсягом до 150,000 запитів за секунду, що призвело до вичерпання пулу з'єднань на рівні веб-серверів.

- **Аналіз відмови**: Базовий ліміт TCP акцепторів був встановлений у 25,000. Процеси-обробники блокували пул Mnesia через повільні клієнтські запити.
- **Вирішення**: Обмеження швидкості (Rate Limiting) перенесено на рівень Istio Ingress Gateway. Пул акцепторів збільшено до 200,000, а парсинг вхідних JSON-пакетів перекладено з Elixir-

інтерпретатора на C99 NIF парсери, що знизило навантаження на CPU з 95% до 12%.

16.6.2 Кейс 2: Розділення мережі (Split-Brain) в Mnesia кластері

Опис інциденту: Внаслідок аварії на комутаторі датацентру виникло розділення мережі між основними нодами кластера, що призвело до паралельного запису даних у різні гілки Mnesia.

- **Аналіз відмови**: Вимкнена автоматична реконсиляція Mnesia призвела до розходження стану таблиць транзакцій.
- **Вирішення**: Налаштовано обробник подій `mnesia_down`. Впроваджено скрипт автоматичного злиття гілок (reconciliation) на основі часових міток транзакцій та перепідключення розділених нод з перезапуском реплікації без втрати даних користувачів.

16.7 Додаток. Інструкція розгортання на Ubuntu 24.04 LTS

У цьому додатку наведено покрокову інструкцію для мінімалістичного розгортання платформи ERP/1 на Ubuntu 24.04 LTS з використанням затвердженого стеку.

16.7.1 Крок 1. Конфігурація та запуск DNS-сервера synrc/ns

DNS-сервер розгортається як інфраструктурний вузол (IP: `10.0.0.5`) для забезпечення локального розв'язання імен домену `erp.uno`.

1. Встановіть середовище виконання Erlang та Elixir:

```
sudo apt update
sudo apt install -y erlang-dev elixir git
```

2. Клонуйте репозиторій та підготуйте проект:

```
git clone https://github.com/synrc/ns.git /opt/synrc-ns
cd /opt/synrc-ns
mix deps.get
```

3. Створіть конфігураційний файл зони `priv/erp.zone.config`:

```
{ttl, 3600}.
{soa, "erp.uno", "ns1.erp.uno", "admin.erp.uno", 2026070101, 86400, 7200,
     604800, 300}.
{ns, "erp.uno", "ns1.erp.uno"}.
{a, "ns1.erp.uno", "10.0.0.5"}.
{a, "k8s-control.erp.uno", "10.0.0.10"}.
{a, "k8s-worker1.erp.uno", "10.0.0.11"}.
```

```
{a, "charts.erp.uno", "10.0.0.5"}.
{a, "argocd.erp.uno", "10.0.0.10"}.
```

4. Відредагуйте `config/config.exs`, вказавши шлях до файлу зони та налаштувавши прослуховування на стандартному порті DNS:

```
import Config
config :ns,
  servers: [
    [{:name, :inet_dns}, {:address, ~c"0.0.0.0"}, {:port, 53}, {:family, :
     inet}]
  ],
  zones: ~c"priv/erp.zone.config"
```

5. Запустіть DNS-сервер в інтерактивній консолі:

```
sudo iex -S mix
```

16.7.2 Крок 2. Налаштування віртуалізації у Proxmox VE

1. Завантажте офіційний образ Ubuntu 24.04 Cloud-Image на хості Proxmox:

```
wget https://cloud-images.ubuntu.com/noble/current/noble-server-cloudimg-
     amd64.img
```

2. Створіть шаблон віртуальної машини (VM ID: 9000):

```
qm create 9000 --memory 2048 --cores 2 --name u24-cloud --net0 virtio,bridge=
    vmbr0
qm importdisk 9000 noble-server-cloudimg-amd64.img local-lvm
qm set 9000 --scsihw virtio-scsi --scsi0 local-lvm:vm-9000-disk-0
qm set 9000 --ide2 local-lvm:cloudinit
qm set 9000 --boot c --bootdisk scsi0
qm set 9000 --serial0 socket --vga serial0
qm template 9000
```

3. Створіть дві ноди кластера шляхом клонування шаблону:

```
qm clone 9000 100 --name k8s-control
qm clone 9000 101 --name k8s-worker1
```

Через інтегляційне Cloud-Init меню вкажіть IP-адреси нод (`10.0.0.10/24` та `10.0.0.11/24`), а як DNS-сервер задайте IP нашого DNS `10.0.0.5`.

16.7.3 Крок 3. Встановлення Kubernetes за допомогою Kubespray

1. Встановіть залежності Ansible на машині управління:

```
sudo apt install -y python3-pip python3-venv git
python3 -m venv venv
source venv/bin/activate
pip install ansible
```

2. Клонуйте Kubespray та встановіть вимоги:

```
git clone https://github.com/kubernetes-sigs/kubespray.git
cd kubespray
pip install -r requirements.txt
```

3. Створіть файл інвентарю `inventory/mycluster/hosts.yaml`:

```
all:
  hosts:
    k8s-control:
      ansible_host: 10.0.0.10
      ip: 10.0.0.10
      access_ip: 10.0.0.10
    k8s-worker1:
      ansible_host: 10.0.0.11
      ip: 10.0.0.11
      access_ip: 10.0.0.11
  children:
    kube_control_plane:
      hosts:
        k8s-control:
    kube_node:
      hosts:
        k8s-control:
        k8s-worker1:
    etcd:
      hosts:
        k8s-control:
    k8s_cluster:
      children:
        kube_control_plane:
        kube_node:
```

4. Запустіть встановлення кластера за допомогою Ansible Playbook:

```
ansible-playbook -i inventory/mycluster/hosts.yaml --become --become-user=
    root cluster.yml
```

16.7.4 Крок 4. Створення та обслуговування Helm-репозиторію

Для збереження автономності розгортаємо мінімалістичний Helm-сервер на базі Nginx на інфраструктурному хості (`10.0.0.5`):

1. Встановіть Nginx та створіть каталог для діаграм:

```
sudo apt install -y nginx
sudo mkdir -p /var/www/helm
```

2. Налаштуйте віртуальний хост Nginx для домену `charts.erp.uno` та перезапустіть службу.

3. Увійдіть до каталогу з вашим Helm-чартом додатку ERP/1 та упакуйте його:

```
helm package ./erp-backend --destination /var/www/helm/
```

4. Згенеруйте індекс репозиторію:

```
helm repo index /var/www/helm/ --url http://charts.erp.uno/
```

16.7.5 Крок 5. Налаштування власного Docker Registry

Для збереження образів контейнерів на інфраструктурному хості (10.0.0.5) розгортається локальний реєстр образів registry.erp.uno.

1. Встановіть інструментарій docker.io та запустіть сервіс реєстру:

```
sudo apt update
sudo apt install -y docker.io
sudo docker run -d -p 5000:5000 --restart=always --name registry registry:2
```

2. Налаштуйте Nginx як реверс-проксі для домену registry.erp.uno з SSL-сертифікатами, згенерованими локальним ca-pki:

```
server {
    listen 443 ssl;
    server_name registry.erp.uno;

    ssl_certificate /etc/nginx/certs/registry.crt;
    ssl_certificate_key /etc/nginx/certs/registry.key;

    location / {
        proxy_pass http://localhost:5000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

3. Зберіть локальний образ додатку на базі Alpine Linux, додайте тег та відправте його до власного реєстру:

```
docker build -t registry.erp.uno/erpuno/crm-documents:1.0.0 .
docker push registry.erp.uno/erpuno/crm-documents:1.0.0
```

16.7.6 Крок 5a. Збірка образів додатків на базі Alpine Linux (Dockerfile)

Для забезпечення стабільної роботи BEAM-релізів у мінімалістичному оточенні Alpine Linux використовується такий багатоетапний (multi-stage) Dockerfile:

```
FROM alpine:3.20 AS builder

RUN apk add --no-cache erlang-dev elixir git build-base

WORKDIR /opt/app

RUN mix local.hex --force && \
    mix local.rebar --force

ENV MIX_ENV=prod
```

```
COPY mix.exs mix.lock ./
COPY config ./config
RUN mix deps.get --only prod && mix deps.compile

COPY lib ./lib
RUN mix release

FROM alpine:3.20

RUN apk add --no-cache openssl ncurses-libs libstdc++

WORKDIR /opt/app

COPY --from=builder /opt/app/_build/prod/rel/erp_system ./

ENV PORT=8080

CMD ["./bin/erp_system", "start"]
```

16.7.7 Крок 6. Налаштування GitOps доставки через ArgoCD

1. Встановіть клієнтський інструментарій `kubectl` на керуючій ноді та підключіть кластер:

```
mkdir -p ~/.kube
sudo cp /etc/kubernetes/admin.conf ~/.kube/config
sudo chown (id -u):(id -g) ~/.kube/config
```

2. Встановіть ArgoCD:

```
kubectl create namespace argocd
kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd
     /stable/manifests/install.yaml
```

3. Створіть файл конфігурації додатку GitOps `erp-app.yaml` для відстеження інфраструктурного репозиторію `helm/`:

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: erp-uno
  namespace: argocd
spec:
  project: default
  source:
    repoURL: 'http://git.erp.uno/erpuno/erp.uno.git'
    path: helm
    targetRevision: HEAD
    helm:
      valueFiles:
        - values.yaml
  destination:
    server: 'https://kubernetes.default.svc'
    namespace: erp
  syncPolicy:
```

```
    automated:
      prune: true
      selfHeal: true
    createNamespace: true
```

4. Налаштуйте Ingress-маршрутизацію для доступу до панелі управління ArgoCD за адресою `argocd.erp.uno` (файл `argocd-ingress.yaml`):

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: argocd-server-ingress
  namespace: argocd
  annotations:
    nginx.ingress.kubernetes.io/ssl-redirect: "true"
    nginx.ingress.kubernetes.io/backend-protocol: "HTTPS"
spec:
  ingressClassName: nginx
  rules:
  - host: argocd.erp.uno
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: argocd-server
            port:
              number: 443
  tls:
  - hosts:
    - argocd.erp.uno
    secretName: argocd-server-tls
```

5. Розгорніть додаток та Ingress в ArgoCD:

```
kubectl apply -f erp-app.yaml
kubectl apply -f argocd-ingress.yaml
```

ArgoCD автоматично виявить зміни в Git-репозиторії, оновить ресурси у кластері та розгорне платформу «ERP/1» у просторі імен `erp`.

16.8 Тестові питання та завдання

1. Які державні інформаційні системи України використовують компоненти ERP/1?

2. Опишіть механізм побудови Erlang-кластера всередині Kubernetes без фіксованих IP-адрес нод.

3. Які переваги дає використання легковагого DNS-сервера `synrc/ns` у порівнянні з класичним BIND/CoreDNS в закритих контурах?

4. Які переваги та структуру мають Helm-діаграми для прикладних продуктів (ІКС) та інфраструктурних служб (SLUB) платформи ERP/1?

5. Опишіть модель асинхронного ETL-процесингу черг з використанням курсорів та контекстів на базі сховища KVS.

6. Які ризики несе явище Split-Brain для розподіленої СУБД Mnesia та як їх мінімізувати?

7. **Практичне завдання**: Напишіть конфігураційний файл зони `synrc/ns` у нативному форматі Erlang Terms для обслуговування трьох нод бази даних Mnesia та однієї ноди ArgoCD.

8. **Практичне завдання**: Напишіть на Erlang функцію ініціалізації контексту воркера `#etl_context{}` з первинним зчитуванням курсора з KVS.

Розділ **17**

Штучний інтелект та інтелектуальні сервіси

17.1 Архітектура локального ШІ-інференсу

У сучасних автономних інформаційних системах (Air-Gapped) використання зовнішніх хмарних моделей ШІ (наприклад, OpenAI або Anthropic) є неприпустимим через вимоги конфіденційності та безпеки. Підсистема «ERP/1: Істотність» базується на використанні локальних компактних моделей ШІ (класу Llama та Phi) на споживчих графічних процесорах (Consumer GPU), що мають лімітований об'єм оперативної пам'яті (до 16 ГБ VRAM).

Архітектура побудована на базі сервісу інференсу `llama.cpp` над операційною системою Ubuntu 24.04 з підтримкою драйверів NVIDIA CUDA 12.x. Інтеграція з основним Erlang/OTP середовищем здійснюється через асинхронні REST API / WebSocket запити за допомогою легковагого HTTP-клієнта.

17.2 Математична модель розрахунку об'єму відеопам'яті (VRAM)

Для забезпечення стабільної паралельної роботи користувачів без ризику виникнення помилок Out-Of-Memory (OOM), здійснюється чітке планування та резервування VRAM. Загальний об'єм відеопам'яті (V_{VRAM}) розподіляється на три ключові компоненти:

$$V_{\text{VRAM}} \geq V_{\text{model}} + V_{\text{KV}} + V_{\text{workspace}} \tag{17.1}$$

Де:

- V_{model} — об'єм пам'яті для зберігання статичних ваг моделі з урахуванням квантування;
- V_{KV} — динамічний буфер пам'яті для зберігання контексту користувачів (Key-Value Cache);

- $V_{\text{workspace}}$ — системний робочий буфер CUDA для виконання обчислень (зазвичай резервується 512–1024 МБ).

17.2.1 Обчислення для моделі Llama-3-8B-Instruct

При квантуванні Q4_K_M статична вага моделі становить $V_{\text{model}} \approx 4.8$ ГБ. Розмір KV-кешу для одного запиту з контекстом $N_{\text{ctx}} = 8192$ токенів розраховується за формулою:

$$V_{\text{KV\_single}} = 2 \times N_{\text{layers}} \times N_{\text{heads}} \times d_{\text{head}} \times N_{\text{ctx}} \times N_{\text{bytes\_per\_elem}} \tag{17.2}$$

Для Llama-3-8B ($N_{\text{layers}} = 32$, $N_{\text{heads}} = 8$, $d_{\text{head}} = 128$, квантований KV-кеш у форматі 16-bit FP):

$$V_{\text{KV\_single}} = 2 \times 32 \times 8 \times 128 \times 8192 \times 2 \approx 1.07 \text{ ГБ} \tag{17.3}$$

На споживчій відеокарті NVIDIA RTX 4070 Ti (12 ГБ VRAM) при $V_{\text{workspace}} = 1.0$ ГБ:

$$V_{\text{KV\_total}} \leq 12.0 - 4.8 - 1.0 = 6.2 \text{ ГБ} \tag{17.4}$$

Максимальна кількість паралельних сесій: $M = \lfloor 6.2/1.07 \rfloor = 5$ користувачів.

17.2.2 Обчислення для моделі Phi-3-medium-14B

При квантуванні Q4_K_M статична вага моделі становить $V_{\text{model}} \approx 8.5$ ГБ. При контексті $N_{\text{ctx}} = 4096$:

$$V_{\text{KV\_single}} = 2 \times 40 \times 10 \times 96 \times 4096 \times 2 \approx 0.625 \text{ ГБ} \tag{17.5}$$

На відеокарті NVIDIA RTX 4070 Ti Super (16 ГБ VRAM) при $V_{\text{workspace}} = 1.0$ ГБ:

$$V_{\text{KV\_total}} \leq 16.0 - 8.5 - 1.0 = 6.5 \text{ ГБ} \tag{17.6}$$

Максимальна кількість паралельних сесій: $M = \lfloor 6.5/0.625 \rfloor = 10$ користувачів.

17.3 Механізми оптимізації інференсу

Для ефективної утилізації обчислювальних ядер GPU застосовуються такі технології:

1. **Continuous Batching (Динамічне пакетування)**: об'єднання запитів на рівні окремих ітерацій генерації токенів, що усуває простої під час виконання фаз prefill (обробка промпту) та decode (генерація наступних токенів).

2. **PagedAttention**: розбиття KV-кешу на невеликі сторінки фіксованого розміру (наприклад, по 16 токенів) та їх динамічний розподіл через віртуальну таблицю вказівок. Це повністю усуває фрагментацію пам'яті та дозволяє спільно використовувати префікси промптів різними користувачами.

17.4 Тестові питання та завдання

1. Чому використання публічних хмарних моделей ШІ є неприпустимим у захищених державних інформаційних системах (Air-Gapped)?

2. Опишіть компоненти загального об'єму відеопам'яті (VRAM), що резервуються для інференсу локальної LLM.

3. Як технологія PagedAttention допомагає боротися з фрагментацією пам'яті GPU?

4. **Практичне завдання**: Розрахуйте об'єм пам'яті для збереження KV-кешу моделі з $N_{layers} = 24$, $N_{heads} = 16$, $d_{head} = 64$ при довжині контексту $N_{ctx} = 2048$ токенів (у 16-бітному форматі).

Розділ 18

Локальні аналітичні сховища (DWH)

18.1 Архітектура локального сховища

Підсистема «ERP/1: Аналітика» призначена для виконання складних аналітичних запитів (OLAP) над даними транзакційних модулів (Mnesia, KVS) без перевантаження основного OLTP-контуру. Основна вимога до підсистеми — робота on-premises, автономність, та повна інтеграція в екосистему Erlang/OTP без використання залежностей від Rust.

Архітектура сховища базується на використанні вбудованої стовпчикової бази даних DuckDB, що працює в ізольованому контурі. Дані з Mnesia трансформуються в оптимізовані стиснені файли Parquet та зберігаються у локальному S3-сумісному об'єктному сховищі (наприклад, Scality).

18.2 DuckDB як вбудований аналітичний рушій

DuckDB підключається безпосередньо до Erlang-вузла через C99 NIF (Native Implemented Functions) обгортку. Для запобігання блокування основних потоків планувальника Erlang (Erlang Schedulers), усі виклики DuckDB виконуються в контурі **Dirty CPU Schedulers**.

18.2.1 Забезпечення ізоляції (Standalone)

Ізоляція DuckDB від транзакційного ядра забезпечується шляхом:

- Запуску DuckDB в окремому OS-процесі або ізольованій групі потоків (Thread Pool) з обмеженням CPU affinity;
- Обмеження виділеної оперативної пам'яті через конфігурацію DuckDB (`SET max_memory = '8GB'`);

- Використання ручного звільнення дескрипторів з'єднань за допомогою функцій очищення ресурсів Erlang NIF (`enif_open_resource_type` та `duckdb_close`).

18.2.2 Горизонтальне масштабування (Scalability)

Оскільки DuckDB є вбудованою однокористувацькою БД, горизонтальне масштабування аналітики реалізується через:

1. **Федеративні SQL-запити (Federated Queries)** — DuckDB здатна читати й об'єднувати гігабайти Parquet-файлів безпосередньо з віддаленого S3 за допомогою розширення `httpfs`. Це дозволяє виконувати розподілені обчислення над єдиним озером даних (Data Lake).
2. **Peer-to-Peer реплікація** — метадані та схеми даних синхронізуються між периферійними аналітичними вузлами через OTP-контур за допомогою легковагого протоколу консенсусу (Raft або Mnesia Sync).

18.3 Порівняльний аналіз аналітичних рішень

У Додатку А викладено порівняльний аналіз трьох рішень для локальних інфраструктур DWH: ClickHouse, MonetDB та Apache Superset.

Критерій	ClickHouse	MonetDB	Apache Supe
Тип системи	Розподілена стовпчикова СКБД	Вбудована / серверна C99 стовпчикова БД	Платформа BI зації звітів
Мова реалізації	C++17 / C++20	Pure C (C99)	Python / React
Рівень складності	Високий (потребує окремого кластера)	Низький (легковажна стовпчикова СКБД)	Середній Python-середо
Інтеграція з Erlang/OTP	Через TCP протокол / HTTP API	Через C NIF або порт зв'язку	Окремий веб- аналітичними

Завдяки реалізації MonetDB на чистому C99, вона є найбільш наближеним та легким закритим конкурентом ClickHouse для впровадження у периферійні вузли ERP/1, які мають жорсткі обмеження на ресурси.

18.4 Тестові питання та завдання

1. Опишіть відмінність між OLTP (Mnesia/KVS) та OLAP (DuckDB) контурами сховища даних ERP/1.

2. Чому при виклику функцій DuckDB через NIF необхідно використовувати саме Dirty CPU Schedulers в Erlang?

3. Завдяки чому реалізується горизонтальне масштабування аналітичних запитів у DuckDB при роботі з Parquet-файлами?

4. Чому СУБД MonetDB є привабливим вибором для локальних аналітичних вузлів з обмеженими ресурсами?

5. **Практичне завдання**: Напишіть SQL-запит для DuckDB, який обчислює сумарний об'єм продажів за категоріями товарів на основі зовнішнього Parquet-файлу, збереженого в об'єктному сховищі S3.

Розділ 19

Безпечний зв'язок та криптографічні протоколи

19.1 Інтеграція та концепція

Підсистема «ERP/1: Комунікатор» об'єднує в єдиний захищений контур сервіси асинхронного обміну миттєвими повідомленнями (CHAT v2) та поштового обміну документами (MAIL v3). Вона абсорбує технічні рішення:

- **chat.n2o.dev** — архітектуру WebSocket-брокерів на базі Erlang/OTP;
- **x509.chat** — автентифікацію за клієнтськими сертифікатами X.509;
- **protocol.zencrypted.uk** — специфікації криптографічного протоколу Buddha (X.422).

19.2 Buddha Protocol (Стандарт X.422)

Протокол Buddha визначає транспортні та криптографічні правила обміну повідомленнями у ненадійних мережах.

19.2.1 X.422.1: Транспортний рівень

Обмін даними здійснюється через асинхронні TCP сокети або шифровані UDP-канали. Для локального виявлення пристроїв у LAN-мережах без DNS-серверів підтримується робота через UDP Multicast на виділений порт 4221. Сервер виконує роль асинхронного брокера, що маршрутизує пакети без збереження їх на сервері після доставки отримувачу.

19.2.2 X.422.2: Криптографічний конверт CMS

Вміст повідомлень та вкладених файлів шифрується на кінцевих пристроях користувачів (End-to-End Encryption) та пакується у

CMS-структуру (RFC 5652) у кодуванні ASN.1 DER:

- **SignedData** — містить підпис ДСТУ 4145 або ECDSA та сертифікат відправника;
- **EnvelopedData** — містить дані, зашифровані симетричним шифром ДСТУ 7624 (Калина) або AES-GCM-256 на сесійному ключі, який зашифрований на відкритому ключі отримувача за схемою ECDH.

19.3 Субкомпоненти контуру безпеки

«Комунікатор» складається з 6 Erlang/OTP додатків:

1. **CA** — локальний центр випуску сертифікатів за протоколами EST (RFC 7030), CMP (RFC 4210), OCSP статусів та TSP міток часу.
2. **VPN** — клієнт-серверне тунелювання WireGuard на базі сертифікатів відкритих ключів.
3. **LDAP** — ієрархічний довідник користувачів з підтримкою DER-кодування та атрибутики ABAC.
4. **IAS** — служба автентифікації сесій та перевірки статусів відкликання ключів користувачів.
5. **CHAT** — асинхронний WebSocket месенджер v2 з підтримкою Double Ratchet.
6. **MAIL** — сервер асинхронної доставки документів та пошти v3 (SMTP/IMAP over TLS).

19.4 Технічна реалізація

19.4.1 Визначення структур даних (Erlang Records)

Основні сутності підсистеми визначені в Erlang-записах:

```
-record(chat_session, {
    session_id,                 % Session UUID (PK)
    client_id,                  % X.509 Certificate Serial Number
    device_id,                  % Device ID
    ip_address,                 % Current IP Address
    access_token,               % Session Access Token
    status = active,            % active | expired | revoked
    created_at = 0
}).

-record(ca_certificate, {
    serial_number,              % X.509 Serial Number (PK)
    subject_dn,                 % Subject Distinguished Name
```

```
    public_key_der,             % Public key in DER encoding
    not_before = 0,             % Not before (start)
    not_after = 0,              % Not after (end)
    status = valid              % valid | revoked | expired
}).
```

19.4.2 Процесний рушій BPE

Для управління життєвим циклом обробки повідомлень використовується BPMN/FSM опис процесу доставки:

```
-module(chat_message).
-include("bpe.hrl").
-compile(export_all).

def() ->
    #process{
        name = 'Buddha Protocol Message Delivery',
        beginEvent = 'ReceiveEnvelope',
        endEvent = 'DestroyMessage',
        tasks = [
            #beginEvent{name='ReceiveEnvelope'},
            #serviceTask{name='RouteByProfile', module=?MODULE},
            #serviceTask{name='StoreTransientQueue', module=?MODULE},
            #serviceTask{name='DeliverToClient', module=?MODULE},
            #serviceTask{name='WaitDeliveryReceipt', module=?MODULE},
            #endEvent{name='DestroyMessage'}
        ],
        flows = [
            #sequenceFlow{name='1', source='ReceiveEnvelope', target='
     RouteByProfile'},
            #sequenceFlow{name='2', source='RouteByProfile', target='
     StoreTransientQueue'},
            #sequenceFlow{name='3', source='StoreTransientQueue', target='
     DeliverToClient'},
            #sequenceFlow{name='4', source='DeliverToClient', target='
     WaitDeliveryReceipt'},
            #sequenceFlow{name='5', source='WaitDeliveryReceipt', target='
     DestroyMessage'}
        ],
        roles = [operator, system]
    }.
```

19.4.3 C99 NIF розбирач TLV DER пакетів

Для швидкого розбору ASN.1 DER об'єктів безпосередньо в Erlang без Cargo/Rust, використовується C99 NIF обгортка:

```
#include "erl_nif.h"
#include <string.h>

static ERL_NIF_TERM parse_tlv_nif(ErlNifEnv* env, int argc, const
     ERL_NIF_TERM argv[]) {
    ErlNifBinary bin;
    if (!enif_inspect_binary(env, argv[0], &bin)) {
```

```
        return enif_make_badarg(env);
    }

    if (bin.size < 2) {
        return enif_make_tuple2(env, enif_make_atom(env, "error"),
                                enif_make_string(env, "too_short",
     ERL_NIF_LATIN1));
    }

    unsigned char tag = bin.data[0];
    unsigned int length = bin.data[1];
    unsigned int header_len = 2;

    if (length & 0x80) {
        unsigned int num_bytes = length & 0x7F;
        if (bin.size < 2 + num_bytes) {
            return enif_make_tuple2(env, enif_make_atom(env, "error"),
                                    enif_make_string(env, "invalid_length",
     ERL_NIF_LATIN1));
        }
        length = 0;
        for (unsigned int i = 0; i < num_bytes; i++) {
            length = (length << 8) | bin.data[2 + i];
        }
        header_len = 2 + num_bytes;
    }

    if (bin.size < header_len + length) {
        return enif_make_tuple2(env, enif_make_atom(env, "error"),
                                enif_make_string(env, "payload_mismatch",
     ERL_NIF_LATIN1));
    }

    ERL_NIF_TERM type_term = enif_make_uint(env, tag);
    unsigned char* payload_data;
    ERL_NIF_TERM payload_term;
    payload_data = enif_make_new_binary(env, length, &payload_term);
    memcpy(payload_data, bin.data + header_len, length);

    return enif_make_tuple3(env, enif_make_atom(env, "ok"), type_term,
     payload_term);
}

static ErlNifFunc nif_funcs[] = {
    {"parse_tlv", 1, parse_tlv_nif}
};

ERL_NIF_INIT(chat_nif, nif_funcs, NULL, NULL, NULL, NULL)
```

19.5 Постквантова криптографія (PQC)

Зі становленням квантових обчислювальних засобів класичні алгоритми асиметричного шифрування (наприклад, RSA, ECDSA, ECDH) та вітчизняний стандарт ДСТУ 4145-2002 стають потенцій-

но вразливими до алгоритму Шора. Для забезпечення довготривалої конфіденційності державного документообігу підсистема «Комунікатор» передбачає адаптаційну інтеграцію постквантових алгоритмів:

- **Кем-механізми (ML-KEM / Kyber)**: Використовуються для безпечного узгодження сесійних симетричних ключів у протоколі Buddha.
- **Схеми підпису на базі решіток (ML-DSA / Dilithium / Falcon)**: Застосовуються для довготривалої автентифікації сертифікатів та засвідчення КЕП документів.
- **Станційні підписи на хеш-деревах (LMS, XMSS)**: Використовуються для підпису образів прошивок VPN-шлюзів та кореневих сертифікатів CA.

19.6 Архітектура нульової довіри (Zero Trust Architecture)

Управління безпекою в ІКС «Комунікатор» базується на концепції Zero Trust (згідно з рекомендаціями NIST SP 800-207). Основні правила побудови периметру безпеки:

1. **Явне підтвердження**: Кожен запит на доступ до реєстрів чи чату проходить обов'язкову автентифікацію на базі клієнтського сертифіката X.509 та авторизацію за політиками ABAC.
2. **Мінімальні привілеї**: Доступ до сервісів надається за принципом найменших повноважень з урахуванням поточного контексту (IP-адреса, робочий час, роль).
3. **Припущення про вразливість**: Усі сегменти мережі вважаються небезпечними. Трафік між сервісами (навіть всередині датацентру) шифрується за протоколом TLS 1.3 або тунелюється через WireGuard.

19.7 Тестові питання та завдання

1. Опишіть призначення та структуру криптографічного конверта CMS протоколу Buddha.
2. Чим відрізняється транспортна схема обміну повідомленнями X.422.1 від класичного HTTPS?
3. Сформулюйте основну перевагу використання C99 NIF для парсингу ASN.1 DER порівняно з парсингом на рівні Erlang.

4. **Практичне завдання**: Напишіть опис структури ASN.1 для найпростішої схеми передачі текстового повідомлення, що містить ідентифікатор сесії та зашифроване тіло.

Розділ 20

LaTeX-орієнтоване управління проєктами

20.1 Концепція продукту «ERP/1: Проєкти»

У великих інженерних проєктах, державних замовленнях та системах подвійного призначення розробка та погодження технічної документації (технічних вимог — ТВ, технічних завдань — ТЗ, технороботчих проєктів — ТРП) є невід'ємною частиною розробки програмного забезпечення.

Популярні інструменти управління проєктами, такі як Atlassian Jira (відстеження задач) та Confluence (база знань), використовують спрощені текстові розмітки (Markdown або Visual Rich Text), які не здатні забезпечити видавничу якість оформлення складних інженерних креслень, блок-схем, специфікацій сокетів та математичних формул.

Модуль «ERP/1: Проєкти» побудований як інтегрована заміна Jira та Confluence, що має такі ключові відмінності:

- **Процесне управління (Jira-аналог)** — робота з Agile-дошками (Scrum, Kanban) та гнучке налаштування статусів задач за допомогою процесного рушія BPMN (BPE).
- **LaTeX як єдиний стандарт розмітки (Confluence-аналог)** — опис вимог до задач та вікі-статей бази знань виконується безпосередньо у форматі LaTeX.
- **Локальне розгортання** — повна автономність та відсутність зовнішніх хмарних залежностей.

20.2 Серверна інтеграція та PDF-компіляція

База знань системи зберігає вихідний код LaTeX-документів у KVS/Mnesia. При кожному оновленні сторінки або за запитом користувача, фоновий Erlang-процес ініціює компіляцію документа за допомогою локального пакету TeX Live (`xelatex` або `pdflatex`).

Скомпільований PDF-документ кешується та відображається безпосередньо в інтерфейсі користувача за допомогою веб-переглядача PDF. Це дозволяє розробникам та технічним письменникам:

1. Використовувати пакет `amsmath` для складних математичних рівнянь.
2. Малювати точні векторні діаграми архітектур та мереж зв'язку за допомогою пакетів `tikz` та `pgf`.
3. Автоматично генерувати готову технічну документацію за стандартами ДСТУ та ISO/IEC 12207 безпосередньо з тикетів та статей бази знань.

20.3 Переваги інтегрованої LaTeX-документації

Впровадження LaTeX-орієнтованого управління проєктами дозволяє:

- **Дотримуватись єдиного джерела істини (Single Source of Truth)**: технічні специфікації, що описують код, зберігаються безпосередньо поруч із кодом у тих самих репозиторіях і мають ідентичний вигляд як у системі ведення завдань, так і в друкованих звітах.
- **Гарантувати відсутність залежностей**: система не потребує сторонніх хмарних конвертерів чи важких JS-бібліотек для рендерингу складних математичних формул на клієнтських пристроях.
- **Спрощувати проходження сертифікації**: згенеровані PDF-файли ТВ та ТЗ повністю відповідають державним вимогам до паперового діловодства, що полегшує здачу робіт замовнику.

20.4 Тестові питання та завдання

1. У чому полягають обмеження розмітки Markdown у системах управління великими інженерними проектами?
2. Опишіть схему взаємодії фонового Erlang-процесу з локальним TeX Live для рендерингу документів бази знань.
3. Які переваги дає використання векторного пакету TikZ для документування архітектури ПЗ?
4. Як інтегрований генератор LaTeX-звітів допомагає при проходженні державної сертифікації програмних рішень?
5. **Практичне завдання**: Напишіть простий шаблон LaTeX-документа для оформлення сторінки технічного завдання, що містить таблицю з переліком вимог до продуктивності системи.

Розділ 21

Структура ІКС ERP/1

21.1 Концепція побудови екосистеми ІКС

Інформаційно-комунікаційні системи (ІКС) платформи «ERP/1» розроблені як комплекс взаємоінтегрованих прикладних рішень для автоматизації державного та комерційного секторів. Усі продукти функціонують на базі єдиного технологічного стеку Erlang/OTP, що гарантує високу масштабованість, стійкість до відмов та безпеку персональних даних. Нижче наведено детальний опис та вимоги до кожного з 10 ключових продуктів ІКС.

21.2 1. Освіта (LMS)

21.2.1 Опис та призначення

Автоматизована інформаційна система управління освітнім процесом, навчальною діяльністю та адмініструванням закладів освіти (згідно з ISO 21001) з підтримкою очної, дистанційної та змішаної форм навчання.

21.2.2 Ключові функціональні модулі

- **Навчальний процес**: електронні курси (SCORM), розклад занять, інтерактивні журнали, та фіксація відвідуваності.
- **Дистанційне навчання**: вбудовані відеоконференції, інтерактивні тести для контролю знань.
- **Контингент**: особові справи учнів та студентів, рух контингенту та академічна мобільність.
- **Кадри та атестація**: особові справи викладачів, облік підвищення кваліфікації, та розрахунок KPI.
- **Інтеграція**: автоматична взаємодія з ЄДЕБО через захищений протокол MQTT та PKI-інфраструктуру.

21.3 2. Здоров'я (HL7)

21.3.1 Опис та призначення

Медична інформаційна система (MIC) для автоматизації медичних послуг та управління медичною інформацією в електронному вигляді згідно з міжнародним стандартом HL7 FHIR.

21.3.2 Ключові функціональні модулі

- **Електронна медична картка (EHR)**: історія хвороб, рецепти, направлення та протоколи лікування.
- **Реєстр пацієнтів**: ведення бази пацієнтів та ідентифікація через демографічні дані.
- **Управління прийомами**: розклад лікарів, електронна черга та онлайн-запис пацієнтів.
- **Взаємодія з eHealth**: двостороння інтеграція з центральною базою даних eHealth України для верифікації декларацій та звітів для НСЗУ.

21.4 3. Документи (CRM)

21.4.1 Опис та призначення

Автоматизація роботи з електронними документами та впровадження електронного документообігу (СЕД) згідно з Державною інструкцією з діловодства №40 від 2024 року.

21.4.2 Ключові функціональні модулі

- **Реєстрація кореспонденції**: вхідні, вихідні, внутрішні документи та звернення громадян.
- **Управління життєвим циклом**: BPE-процеси погодження, підписання (КЕП) та накладання резолюцій.
- **Сховище документів**: організація ієрархічної структури папок на базі KVS/Mnesia з контролем доступу ABAC.

21.5 4. Облік (ACC)

21.5.1 Опис та призначення

Комплексна автоматизація бухгалтерського обліку, кадрового діловодства та розрахунку заробітної плати для державних установ та приватних підприємств відповідно до законодавства України.

21.5.2 Ключові функціональні модулі

- **Головна книга**: план рахунків, подвійний запис, облік ТМЦ, ОЗ, ПДВ, розрахунки з контрагентами.
- **Кадровий облік**: штатні розписи, накази по персоналу, особові картки П-2, та табель обліку робочого часу.
- **Заробітна плата**: нарахування окладів, премій, лікарняних, розрахунок ЄСВ та ПДФО, генерація податкової звітності.

21.6 5. Склад (WMS)

21.6.1 Опис та призначення

Автоматизована система управління складським господарством, матеріально-технічним забезпеченням та логістичними ланцюгами постачання.

21.6.2 Ключові функціональні модулі

- **Адресне зберігання**: динамічний розподіл складських зон, стелажів та осередків.
- **Операційний облік**: приймання, переміщення, комплектація, відвантаження ТМЦ та інвентаризація.
- **Штрихкодування**: інтеграція з терміналами збору даних (ТЗД) та підтримка кодів маркування GS1.

21.7 6. Реєстри (CART)

21.7.1 Опис та призначення

Універсальна low-code платформа для створення, ведення та інтеграції державних, відомчих чи корпоративних інформаційних реєстрів будь-якого масштабу.

21.7.2 Ключові функціональні модулі

- **Конструктор схем**: візуальне проектування схем даних та автоматична генерація таблиць у сховищі.
- **Трасування змін**: фіксація кожного запису у вигляді незмінного ланцюжка транзакцій з накладанням КЕП.
- **Шина Трембіта**: вбудований адаптер для взаємодії з національним простором електронної взаємодії.

21.8 7. Істотність (AI)

21.8.1 Опис та призначення

Локальна ШІ-інфраструктура для інтелектуального аналізу, семантичного пошуку та автоматичної оцінки істотності інформації у документах та реєстрах без вивантаження даних у хмару.

21.8.2 Ключові функціональні модулі

- **Семантичний аналіз**: класифікація документів, пошук за змістом та виявлення суперечностей.
- **CUDA-інференс**: робота з моделями Llama-3 (8B) та Phi-3 (14B) на споживчих GPU (16GB VRAM) за технологіями PagedAttention та Continuous Batching.

21.9 8. Аналітика (DWH/OLAP)

21.9.1 Опис та призначення

Високопродуктивна аналітична інфраструктура DWH для векторизованої обробки великих обсягів даних on-premises на базі вбудованого рушія DuckDB/MonetDB.

21.9.2 Ключові функціональні модулі

- **Обробка Parquet**: генерація стовпчикових звітів безпосередньо з локального S3 сховища.
- **C99 NIF інтеграція**: швидке виконання OLAP запитів через Dirty CPU Schedulers планувальника Erlang.

21.10 9. Проєкти (PM)

21.10.1 Опис та призначення

Локальна заміна Atlassian Jira та Confluence на базі Erlang/OTP, розроблена для управління життєвим циклом проектів розробки ПЗ та ведення баз знань.

21.10.2 Ключові функціональні модулі

- **Agile-управління**: Scrum/Kanban дошки, спринти, беклоги та BPE BPMN workflow для задач.
- **LaTeX Wiki**: ведення всієї документації проекту у форматі LaTeX з автоматичною генерацією ТВ, ТЗ та ТРП видавничої якості у PDF.

21.11 10. Інциденти (ITSM)

21.11.1 Опис та призначення

Система управління IT-послугами (ITSM) відповідно до рекомендацій бібліотеки найкращих практик ITIL для реєстрації та вирішення інцидентів, запитів та проблем.

21.11.2 Ключові функціональні модулі

- **Service Desk**: прийом та класифікація звернень користувачів.
- **SLA-контроль**: автоматичне відстеження часових лімітів вирішення інцидентів.
- **База знань відомих помилок (KEDB)**: інтеграція з Вікі для швидкого вирішення типових збоїв.

21.12 Тестові питання та завдання

1. Перелічіть 10 основних прикладних продуктів (ІКС) платформи ERP/1.
2. У чому різниця між ІКС-продуктами та інфраструктурними службами ERP/1?
3. Які криптографічні механізми використовуються для підтвердження юридичної значущості в ІКС «Документи»?
4. Опишіть призначення бази знань відомих помилок (KEDB) в ІКС «Інциденти» (ITSM).
5. **Практичне завдання**: Намалюйте BPMN/FSM діаграму переходів статусів інциденту від моменту його реєстрації в Service Desk до закриття користувачем.

Розділ **22**

Висновки

Підбиваючи підсумки нашого дослідження та практичного досвіду впровадження національних інформаційних систем, неможливо оминути ключовий фактор, який забезпечив беззаперечний успіх наших наймасштабніших проєктів — це свідомий і технологічно вивірений вибір платформи. У той час як більшість інших державних відомств та міністерств продовжують йти консервативним шляхом, обираючи громіздкі екосистеми на базі CLR (.NET) та JVM (Java Virtual Machine), ми з неймовірною бадьорістю, ентузіазмом і професійним запалом розгорнули ключові системи держави на фундаменті промислових телекомунікаційних технологій Ericsson Erlang/OTP.

Цей архітектурний вибір був продиктований суворими вимогами до відмовостійкості. Технології Erlang/OTP, створені для безперервної роботи в умовах пікових навантажень, дозволили нам у рекордно короткі терміни та з мінімальними операційними витратами імплементувати такі колосальні системи, як:

- **Депозити АТ КБ «ПриватБанк»** — серцевина найбільшого банку країни, що бездоганно і стабільно гарантує консистентність мільйонів фінансових транзакцій і збереження активів нації;
- **СЕД «МІА: Документообіг» для МВС України** — надійна і криптографічно захищена інформаційна система, яка забезпечила безперебійний обіг мільйонів організаційно-розпорядчих документів та кримінальних проваджень у найбільшому силовому відомстві;
- **СЕД «ERP/1: Документи» для МО України** — система захищеного документообігу та автоматизації діловодства для забезпечення життєдіяльності Збройних Сил України;
- **ЕСОЗ для НСЗУ та МОЗ** — центральний компонент медичної реформи європейського зразка, що витримує високоінтенсивну обробку чутливих медичних даних мільйонів громадян України в режимі реального часу;

- **Модернізовані модулі ERP/1** — інтелектуальний аналіз документів на базі локального ШІ («Істотність»), вбудоване масштабоване сховище («Аналітика»), наскрізне криптографічне шифрування за протоколом Buddha X.422 («Комунікатор»), та гнучке управління задачами з LaTeX-базою знань («Проєкти»).

Контраст результатів вийшов вичерпним і очевидним. Замість того, щоб щороку змушувати державу витрачати сотні мільйонів та колосальні ресурси на нескінченні міграції, ліцензійні відрахування та підтримку актуальних версій і фреймворків (для латання архітектурних дір важких enterprise-рішень на базі Java чи C#), проєкти на Erlang/OTP продемонстрували, якою елегантною, економною і граційною може бути надскладна національна IT-інфраструктура. Завдяки ізоляції процесів, вбудованій філософії уникнення збоїв («let it crash») та унікальній механіці гарячого оновлення коду («hot code swapping») без зупинки роботи, ці системи здатні функціонувати роками без жодних суттєвих перезавантажень, заощаджуючи величезні кошти платників податків.

Ми довели на світовій практиці, що створення електронної держави — це не обов'язково бюрократичний довгобуд із постійним роздуванням бюджетів і штатів програмістів. З правильною архітектурною візією та надійними функціональними інструментами, цей процес перетворюється на швидке, веселе і драйвове інженерне мистецтво високого ґатунку, здатне вивести управління державними та банківськими активами на безпрецедентний рівень якості.

Розділ **23**

Словник термінів та скорочень

23.1 Список скорочень

ВККС — Вища кваліфікаційна комісія суддів України.

ВРП — Вища рада правосуддя.

ГРД — Громадська рада доброчесності.

ГРМЕ — Громадська рада міжнародних експертів.

ГСЦ — Головний сервісний центр МВС.

ДМС — Державна міграційна служба України.

ДПС — Державна прикордонна служба України.

ДСА — Державна судова адміністрація.

ДСНС — Державна служба України з надзвичайних ситуацій.

ЕСОЗ — Електронна система охорони здоров'я МОЗ.

ЄІС — Єдина інформаційна система МВС.

ЄРЗ — Єдиний реєстр зброї НПУ.

ЄСІКС — Єдина судова інформаційно-комунікаційна система.

КЕП — Кваліфікований електронний підпис.

КЗІ — Криптографічний захист інформації.

КМУ — Кабінет Міністрів України.

КСЗІ — Комплексна система захисту інформації.

МВС — Міністерство внутрішніх справ України.

МОЗ — Міністерство охорони здоров'я України.

МОН — Міністерство освіти і науки України.

МОУ — Міністерство оборони України.

НАН — Національна академія наук України.

НГУ — Національна гвардія України.

НДІ — Науково-дослідний інститут.

НПА — Нормативно-правовий акт.

НПУ — Національна поліція України.

ОВВ — Органи виконавчої влади.

ПФУ — Пенсійний фонд України.

ССО — Служба судової охорони / Сили спеціальних операцій.

СУСЗЦЗ — Система управління силами та засобами цивільного захисту ДСНС.

ТЗІ — Технічний захист інформації.

СТЗІ — Система технічного захисту інформації.

ФП МТРЗ — Функціональна підсистема матеріально-технічного та ресурсного забезпечення МВС.

ЦГЗ — Центр громадського здоров'я.

ЦОВВ — Центральний орган виконавчої влади.

ЦОД — Центр обробки даних.

23.2 Словник термінів (Глосарій)

ABAC (Attribute-Based Access Control) — Управління доступом на основі атрибутів. Модель безпеки, де рішення про надання доступу приймається шляхом зіставлення властивостей (атрибутів) суб'єкта, об'єкта та поточного контексту (наприклад, часу чи IP-адреси).

BEAM — Віртуальна машина мов програмування Erlang та Elixir. Забезпечує паралельну обробку мільйонів ізольованих легковагових процесів з високою відмовостійкістю та автоматичним збиранням сміття.

BPE (Business Process Engine) — Рушій бізнес-процесів. Компонент екосистеми ERP/1, призначений для управління життєвим циклом об'єктів та виконання переходів станів відповідно до моделей BPMN.

CA (Certificate Authority) — Центр сертифікації. Додаток інфраструктури відкритих ключів (PKI), який здійснює випуск, відкликання та перевірку статусів цифрових сертифікатів X.509.

CMS (Cryptographic Message Syntax) — Синтаксис криптографічних повідомлень. Стандарт IETF (RFC 5652) для криптографічного захисту повідомлень (підпис, шифрування, конвертування) у двійковому форматі ASN.1.

DER (Distinguished Encoding Rules) — Унікальні (канонічні) правила кодування. Стандарт серіалізації структур ASN.1, який гарантує єдине бієктивне представлення даних на фізичному рівні, запобігаючи атак на розходження парсерів.

DWH (Data Warehouse) — Сховище даних. Спеціалізована база даних, оптимізована для агрегації та аналізу великих масивів інформації з різних джерел без перевантаження OLTP-систем.

eIDAS (electronic IDentification, Authentication and trust Services) — Регламент Європейського Союзу про електронну ідентифікацію та довірчі послуги для електронних транзакцій на внутрішньому ринку.

EUDI Wallet (European Digital Identity Wallet) — Європейський гаманець цифрової ідентифікації. Програмне забезпечення для смартфонів, яке дозволяє громадянам безпечно зберігати, пред'являти та обмінюватися даними про особу (PID) та інші цифрові атестати атрибутів.

GDPR (General Data Protection Regulation) — Загальний регламент про захист даних. Регламент Європейського Союзу

щодо захисту персональних даних усіх осіб у межах Європейського Союзу та Європейської економічної зони.

KVS (Key-Value Storage) — Вбудоване сховище типу «ключ-значення» в екосистемі ERP/1, що використовує ієрархічне індексування та підтримує незмінність (immutability) даних.

Mnesia — Вбудована розподілена СУБД реального часу для віртуальної машини BEAM. Підтримує реплікацію та транзакції ACID над оперативною пам'яттю та диском.

NIF (Native Implemented Function) — Функція, написана на іншій мові програмування (зазвичай C/C++ або Rust) і скомпільована як бібліотека спільного доступу для безпосереднього виклику з віртуальної машини Erlang (BEAM).

OLAP (Online Analytical Processing) — Аналітична обробка в реальному часі. Метод обробки даних, орієнтований на виконання складних аналітичних запитів (наприклад, DuckDB).

OLTP (Online Transaction Processing) — Транзакційна обробка в реальному часі. Метод обробки даних, орієнтований на часті та швидкі транзакції запису й оновлення (наприклад, Mnesia).

PQC (Post-Quantum Cryptography) — Постквантова криптографія. Галузь криптографії, яка розробляє алгоритми шифрування та підпису, стійкі до компрометації за допомогою квантових комп'ютерів.

SLA (Service Level Agreement) — Угода про рівень послуг. Офіційний договір, який фіксує вимоги до якості та швидкості надання послуг (наприклад, SLA для вирішення інцидентів у ITSM).

Zero Trust (Нульова довіра) — Безпекова модель, за якої жоден пристрій чи користувач не вважаються надійними за замовчуванням (навіть у внутрішній локальній мережі) і проходять перевірку доступу при кожному запиті.

Предметний покажчик