[Вступ 1](#_Toc1272202021)

[Підсумковий звіт 2](#_Toc1630690861)

[Предмет аудиту 3](#_Toc584277536)

[Бізнес-процеси 3](#_Toc1524948154)

[Форми 3](#_Toc1327548952)

[Аудит бізнес-процесів 5](#_Toc1852774689)

[BP-01. X-Access-Token в сервісних задач 5](#_Toc1209985214)

[BP-03. Транзакції в циклах 6](#_Toc719670028)

[BP-05. Ліміт для критеріїв пошуку 7](#_Toc1564376853)

[BP-08. Декілька викликів фабрики даних в одній транзакції 8](#_Toc385633590)

[BP-09. Ініціалізація та використання змінних 9](#_Toc1194900128)

[BP-11. Створення читабельних BPMN діаграм 12](#_Toc1488496267)

[BP-15. Таймери на користувацьких задачах 12](#_Toc944362810)

[BP-18. Історичні події для високонавантажених бізнес-процесів 13](#_Toc1907316253)

[BP-19. Інше 14](#_Toc530522336)

[BP-19-01. Бізнес-процеси розробників 14](#_Toc1968896533)

[Аудит моделі даних 14](#_Toc1656759628)

[DM-01. Індекси для критеріїв пошуку 14](#_Toc1422162006)

[DM-04. Перелік колонок в критеріях пошуку 15](#_Toc968927089)

[DM-05. Ліміти на критерії пошуку 16](#_Toc2135452668)

[DM-06. Нормалізація схеми бази даних 17](#_Toc1831364035)

[DM-06-1 Відсутність унікальних ключів для полів таблиць, які повинні мати унікальні значення. 18](#_Toc1136337019)

[DM-06-2 Повна відсутність зовнішніх ключів (Foreign Keys, FK) у базі даних. 18](#_Toc1168860215)

[DM-06-03 Використання JSON-полів для зберігання структурованих даних 19](#_Toc155137187)

[DM-09. Пагінація (Посторінкова навігація) на критеріях пошуку 19](#_Toc1982604619)

[DM-09. Інше 21](#_Toc76786262)

[DM-09-01 Критерій пошуку, що дістає всі дрфо користувачів 21](#_Toc1093272005)

[Аудит UI-форм бізнес-процесів 22](#_Toc480509761)

[FR-01. Великі форми 22](#_Toc1161464333)

[FR-02. Lazy load для компонента Select 23](#_Toc386018778)

[FR-03. Javascript логіка в компонентах форми 23](#_Toc938531768)

[FR-06. Експериментальні компоненти 25](#_Toc807271674)

[FR-07. Ліміти на критеріях пошуку для компонента Select 26](#_Toc120880573)

[Аудит безпеки 26](#_Toc1482316332)

[SC-01. Принцип найменших привілеїв 26](#_Toc1726946830)

[SC-02. Мінімізація ролей з розширеними правами 26](#_Toc801148278)

[SC-03. Принцип розділення обов’язків 27](#_Toc2123126191)

[SC-04. RLS (Row Level Security) на моделі даних 27](#_Toc271858730)

[SC-05. RBAC (Role Based Access Control) на моделі даних 28](#_Toc693564875)

[SC-06. Конфіденційні дані 29](#_Toc793786115)

# Вступ

В рамках проведення аудиту регламенту реєстру X був взятий за основу документ з загальними рекомендаціями по аудиту та розробці - [посилання.](https://diia-engine.github.io/diia-engine-documentation/ua/platform/1.9.7/registry-develop/registry-audit-instruction/registry-audit-instruction.html) У зв’язку з обмеженням людського і часового ресурси скоуп робіт по бізнес-процесам і формах був обмежений переліком, який є доступний кінцевому користувачу і оцінений, як найбільш навантажений. Перелік формувався на основі авторизаційної моделі бізнес-процесів. Були виключені адміністративні, бізнес-процеси розробників та ті, які доступні представникам міністерства.

Аудит проводився в 2 етапи:

* Версія регламенту 11.11.23
* Версія регламенту 12.12.24

Частина знахідок і рекомендацій постійно передавались команді розробки протягом проведення аудиту і були виправлені у другій версії регламенту, що підлягала аудиту.

Перелік місць по знахідкам не є вичерпним а представляє загальний шаблон використання, який повинен бути повністю усунутий.

## Підсумковий звіт

Кількість знахідок по категоріям і критичності

|  |  |  |  |
| --- | --- | --- | --- |
|  | Висока | Середня | Низька |
| Продуктивність | 8 | 0 | 1 |
| Надійність | 1 | 3 | 0 |
| Безпека | 7 | 1 | 0 |
| Підтримуваність | 1 | 3 | 0 |

З загальних шаблонів, які зустрічаються в регламенті, можна виділити наступні найбільш критичні:

* Вивантаження всієї таблиці для передзаповнення даних на формі
* Відсутність лімітів, фільтрів та пагінації на критеріях пошуку
* Використання важких персистентних змінних бізнес-процесів
* Відсутність належного процесу підготовки регламенту до промислового середовища і видалення всього зайвого (демо ролі, демо шматки коду, процеси, що ще в розробці)
* Відсутність індексів на базі даних
* Відсутність розділення доступу до даних
* Відсутність належних правил роботи з конфіденційними даними
* Фактично кастомна розробка форм, що може бути проблемою при оновленні версії платформи і вимагати додаткової розробки

Предмет аудиту

#### Бізнес-процеси

1. medic-log
2. team-lead-control
3. registry-doc-arrange
4. entity-summary
5. protocol-records
6. hr-supervise-all
7. group-access-queue
8. account-verify
9. client-info
10. profile-details

#### Форми

1. adjust-schedule
2. adjust-schedule-auth
3. new-order-summary-final
4. new-order-summary-final-edit
5. new-order-summary-final-edit-auth
6. new-order-summary-final-auth
7. new-order-summary-step-i
8. new-order-summary-step-ii
9. new-order-summary-step-iii
10. new-order-summary-step-iv-opt
11. new-order-summary-step-v
12. new-order-summary-step-vi
13. new-order-summary-v2-final
14. new-order-summary-v2-final-edit
15. new-order-summary-v2-final-edit-auth
16. new-order-summary-v2-final-auth
17. new-order-summary-v2-step-ii
18. new-order-summary-v2-step-iii
19. new-order-summary-v2-step-iv
20. new-order-summary-v2-step-vi
21. new-order-summary-v2-step-vi-edit
22. new-order-summary-v2-step-vi-edit-auth
23. new-order-log
24. new-order-v2-log
25. patient-issue
26. snippet-alert

# Аудит бізнес-процесів

## BP-01. X-Access-Token в сервісних задач

Критичність: середня

Категорія: надійність

Вплив:

Термін дії користувацького авторизаційного токена може закінчитися, поки бізнес-процес дійде до точки використання. Може бути не виявлено на етапі розробки та тестування через відсутність відповідних умов (задача, яку не беруть у виконання понад 5 хв, довга транзакція, тимчасова недоступність сервісу тощо)

Опис:

При використанні *juel* функцій для отримання користувацького авторизаційного токену треба впевнитись, що місце використання функції та відповідна актівіті (користувацька задача, або початкова подія) знаходяться в одній транзакції бізнес-процесу. Також збереження токену в сталій (non-transient) не є безпечним оскільки кожну сталу змінну можна побачити в Веб-інтерфейсі управління виконанням бізнес-процесів. Також збереження токену в сталій змінній може призвести до використання неактуального токену в сервісних задачах (докладніше в розділі BP-09)

Де знайдено:

* protocol-records / Підписати та зберегти дані (activitySign) - після цієї користувацької задачі не виконується оновлення токена у змінній *initAccToken*, тому виклик декількох сервісних задач в кінці бізнес-процесу використорвують токен попередньої транзакції.

Рекомендації: Додати скрипт що оновлює змінну *initAccToken* після Підписати та зберегти дані (activitySign)
* hr-supervise-all / Ініціалізація accessToken (Activity\_0y1ra7k) - в цій скрипт-тасці використовується токен не останньої задачі - Підпис КЕП/УЕП (activitySign) а попередньої - Перегляд всіх елементів довідника (activityCreateTimeslots), тому виклик декількох сервісних задач в кінці бізнес-процесу використорвують токен попередньої транзакції.

Рекомендації: Змінити ініціалізацію *initAccToken* з токеном completer('activitySign').accessToken
* group-schedule / Перегляд/редагування даних (activityTerminated) - після цієї користувацької задачі не виконується оновлення токена у змінній *initAccToken,* тому коли бізнес-процес повертається на activityQuestionnairePage2 то запити на довідники до задачі - Перегляд/редагування даних (activityQuestionnairePage2) можуть виконуватися з неактуальним токеном

Рекомендації: Додати скрипт що оновлює змінну *initAccToken* після Перегляд/редагування даних (activityTerminated)
* Реєстрація користувача в системі (user-confirm-registration) / Інформація привітання (activityWelcome) - після цієї користувацької задачі не виконується оновлення токена у змінній *initAccToken,* тому наступна сервісна задача Отримати перелік компаній з БД (Activity\_1pvg0nd) використовує неактуальний токен користувача

Рекомендації: Додати скрипт що оновлює змінну *initAccToken* після Інформація привітання (activityWelcome)

## BP-03. Транзакції в циклах

Критичність: середня

Категорія: надійність

Вплив:

При наявності циклів, які мають велику кількість ітерацій Camunda Engine тримає транзакцію на базі даних протягом всього часу його виконання, що може призвести до впливу довгих транзакцій. Додатково слід зауважити, що при виникненні помилки на 100-ій ітерації, буде спроба виконати всі ітерації, починаючи з першої.

Опис:

Причиною довгої транзакції може бути цикл, який виконує велику кількість ітерації, тому при моделюванні циклів в бізнес-процесах необхідно застосовувати асинхронне виконання для кожної ітерації.

Де знайдено:

* group-access-queue / Призначення керівника (Group\_00mue9k): update-teams-boss ресурс оновлюється в циклі: що може призвести до довгої транзакції

Рекомендації:

* Розглянути використання асинхронного виконання циклу - проставити *camunda:asyncBefore* атрибут на Підготовка циклу збереження (Activity\_12kr5gc) та використати system\_user().accessToken на всіх сервісних задачах між цим скриптом та наступною користувацькою задачею. Також можна використати елемент Expanded Sub Process з налаштованим Multi-instance для оновлення update-teams-boss.

## BP-05. Ліміт для критеріїв пошуку

Критичність: висока

Категорія: продуктивність

Опис:

При використанні сервісної задачі з пошуку сутностей в фабриці даних або виклику АПІ Кейклоака треба явно задавати параметр по максимальній кількості даних (limit), які можуть бути отримані.

* Збільшує навантаження на базу даних (Кейклоака та реєстра). Довго тримає конекшен до бази, інші запити стають в чергу, збільшує загальний час виклику по сервісам
* Збільшує використання хіпа по ряду сервісів (keycloak, bpms, registry-rest-api). Довго тримає конекшен до бази та тред з тред пула, інші запити стають в чергу, збільшує загальний час виклику по сервісам.

Приклад використання:

* group-access-queue / Отримати дані довідника з БД (Activity\_05irp75). - виклик ресурсу get-organizations без критеріїв пошуку, лімітів та пагінації
* profile-details / Отримати перелік всіх користувачив з keycloak (Activity\_198yrv5) - цей виклик вивантажує всіх користувачів в бізнес-процес для подальшого опрацювання
* client-info (первинна версія регламенту) / Отримати дані довідника з БД (Activity\_07isovq) - має фільтрацію по двум параметрам типу boolean (teamActive та isAssigned), що потенційно вивантажує ¼ всіх даних у відповідній таблиці
* medic-log (первинна версія регламенту) / Отримати дані по дігнозам (Activity\_1y9fo2j) - має фільтрацію по параметру типу boolean (directoryMkxActive) що потенційно вивантажує 50% даних таблиці в бізнес-процесу
* entity-summary/ Отримати Організації з БД (Activity\_1owk9w8) - вивантажує всі дані ресурсу get-organizations без критеріїв пошуку, лімітів та пагінації

Рекомендації:

* Не вивантажувати всі дані по таблиці реєстру в пам’ять бізнес-процесу для підготовки форми, а підтягувати відповідні дані на формі через селекти з критеріїв пошуку фабрики даних з використанням параметрів фільтрації та лімітів
* В бізнес-процесі при необхідності опрацювання всіх користувачів:
	+ Робити вивантаження пачками з пагінацією і поетапним опрацюванням
	+ Переглянути необхідність вивантаження всіх значень таблиці. Застосувати фільтрацію по певному критерію

## BP-08. Декілька викликів фабрики даних в одній транзакції

Критичність: середня

Категорія: надійність

Вплив:

Створення неконсистентних даних в базі даних після виникнення помилки між окремими викликами фабрики даних. В залежності від логіки та моделі регламенту може призвести до повного блокування роботи з конкретним записом.

При виникненні помилки, політика повторних спроб виконання бізнес-процесу розпочне виконання з початку, що може призвести до повторної вставки даних в окрему таблицю.

Опис:

При моделюванні бізнес-процесу може виникнути необхідність оновлення декількох таблиць бази даних в рамках однієї транзакції (бази даних, не плутати з транзакцією бізнес-процесу). Тобто щоб або всі таблиці були оновлені, або жодна з них. На рівні виконання бізнес-процесу не має можливості пов’язати декілька викликів фабрики даних в одну транзакцію, тому декілька послідовних викликів фабрики даних в одному бізнес-процесі можуть призвести до створення неконсистентних даних в базі даних.

Де знайдено:

* account-verify / Перегляд/редагування даних (activityBlankFinalSign) - після цієї задачі виконується декілька збережень в дата-фабрику що є не ідемпотентною операцією

Рекомендації:

* Оскільки використання nested entity неможливе через наявність апдейт операцій в транзакції бізнес-процесу, тому рекомендується смоделювати компенсаційні дії на ці операції, та змоделювати операції апдейту бази якомога щільіше

## BP-09. Ініціалізація та використання змінних

Критичність: висока

Категорія: продуктивність, безпека

Вплив:

* Збереження токена користувача в сталій змінній може призвести до проблем з безпекою, бо всі сталі змінні зберігаються в базі даних Camunda і їх можна побачити та навіть змінювати в Веб-інтерфейсі управління виконанням бізнес-процесів.
* Також велика кількість масивних сталих змінних збільшує навантаження на базу даних Camunda, що в свою чергу збільшує час транзакції бізнес-процесу.

Опис:

При необхідності створення додаткової змінної в бізнес-процесі ініціалізувати її якомога ближче до місця використання. Також необхідно мінімізувати кількість та розмір сталих змінних.

Використання токена в сталій змінній:

* hr-supervise-all - збереження токена користувача як сталу (non-transient) змінну
	+ Ініциалізація accessToken, bpTimeOut (Activity\_0q7yhor)
	+ Зміна токена (Activity\_1ddy2pg)
	+ Ініціалізація accessToken (Activity\_0ektqic)
	+ Ініціалізація accessToken (Activity\_1ssn51o)
	+ Ініціалізація accessToken (Activity\_1494y0d)
	+ Ініціалізація accessToken (Activity\_1esrxmp)
	+ Ініціалізація accessToken (Activity\_029b6tq)
	+ Ініціалізація accessToken (Activity\_0jayguo)
	+ Ініціалізація accessToken (Activity\_1cppcvk)
	+ Ініціалізація accessToken (Activity\_1dfgyuq)
	+ Ініціалізація accessToken (Activity\_0ura22x)
	+ Ініціалізація accessToken (Activity\_0vxzu5t)
	+ Ініціалізація accessToken (Activity\_0dv53k2)
	+ Ініціалізація accessToken (Activity\_1jhg5zu)
* team-lead-control - збереження токена користувача як сталу (non-transient) змінну
	+ Ініциалізація accessToken та timeOut процесу (Activity\_0q7yhor)
	+ Ініціалізація accessToken (Activity\_0ektqic)
	+ Ініціалізація accessToken (Activity\_1ssn51o)
	+ Ініціалізація accessToken (Activity\_1k45sut)

Рекомендації:

* Використовувати для збереження токену користувача несталу (transient) змінну

Ініціалізація змінної задовго до її використання:

* client-info / Підготовка даних для відображення (Activity\_1biwds4) - скрипт має назву Підготовка даних для відображення хоча дані які там формуються не відображаються на наступній формі, ці дані використувоються вже після Журнал справ (activityDictMain) хоча ініціалізуються до неї

Рекомендації:

* Ініціалізувати змінну безпосередньо перед її використанням. Під ініціалізацією змінної може бути як і явне її створення, так і використання будь-яких задач, результат яких також зберігається як змінна.

## BP-11. Створення читабельних BPMN діаграм

Критичність: низька

Категорія: підтримуваність

Опис:
При моделюванні BPMN діаграм використовувати загально прийняті практики, деталі яких можна знайти за [посиланням](https://diia-engine.github.io/diia-engine-documentation/ua/platform/1.9.7/registry-develop/registry-audit-instruction/modules/bp-audit.html#_bp_11)

Де знайдено:

* account-verify / в тегу опису bpmn:collaboration використовується невалідний атрибут, camunda:delegateExpression="${keycloakGetUserRoleConnectorDelegate}", який скоріш за все був доданий помилково при копіюванні частини бізнес-процесу

Рекомендації:

Видалити зайві атрибути з тегів опису бізнес-процесів (xml код бізнес-процесу)

Створення читабельних потоків послідовностей (sequence flows):

* account-verify / Розгалуження після init (Activity\_0ganl0c) не має імені тому не зрозуміло що повиннен зробити користувач щоб потрапити до нижньої гілки. Також ми маємо 2 розгалуження з імʼям “тест page 2?“ проте вони мають різні кондішени.
* hr-supervise-all / Розгалуження після init (Activity\_0ganl0c) не має імені
* hr-supervise-all / Ініціалізація accessToken (Activity\_08x5y9z) - скрипт має назву Ініціалізація accessToken, проте виконує іншу операцію не повʼязану з токенами.
* account-verify / зайві елементи або ланцюжки елементів що не використовуються в бізнес-процессі.

Рекомендації:

* Іменувати всі елементи бізнес-процесу відповідно до виконуваної операції або перевірки на розгалуженнях.
* Видалити зайві елементи що не використовуються бізнес-процесом або додати коментарі якщо ці елементи потрібні і чому.

## BP-15. Таймери на користувацьких задачах

Критичність: висока

Категорія: продуктивність

Вплив:

Велике накопичення відкритих бізнес-процесів через користувацькі задачі, які вже не будуть виконані призводить до безпотрібного навантаження на систему та використання її ресурсів та необхідності виконання додаткових операцій по видаленню запущених бізнес-процесів

Опис:

При роботі з бізнес-процесами реєстру відповідальні особи працюють з користувацькими задачами (user task), які були на них призначені і які з тих чи інших причин можуть бути виконані та забуті. Одним з можливих рішень є використання таймерів з автоматичним завершенням бізнес-процесу

Де знайдено:

* account-verify / Повідомлення суб'єкта (activityMessage)
* account-verify / Повідомлення суб'єкта (Activity\_0owgxfe)
* account-verify / Повідомлення суб'єкта (Activity\_1dj9zzs)
* account-verify/ Повідомлення суб'єкта (Activity\_0p6bmdi)
* egistry-doc-arrange / Інфо про завершення (Activity\_17zat1v)

## BP-18. Історичні події для високонавантажених бізнес-процесів

Критичність: низька

Категорія: продуктивність

Вплив:

Додаткове навантаження на Підсистему асинхронного обміну повідомленнями та Сервіс фіксації історичних подій через велику кількість історичних подій в процесі виконання бізнес-процесів, що призводить до збільшення часу затримок і відмови окремих компонентів системи

Додаткове навантаження на Підсистему управління реляційними базами даних, що можуть призвести до відмови ключових сервісів, таких як Сервіс виконання бізнес-процесів, що практично повністю блокує роботу реєстру

Опис:

При виконанні бізнес-процесів для нього зберігаються історичні події пов’язані з його проходженням, такі як виконання задачі, зберігання окремих видів змінних тощо. Процес зберігання історичних подій може суттєво навантажувати систему і навіть бути причиною помилок, якщо кількість таких процесів є досить великою. Рекомендується відповідним чином адаптувати бізнес-процес, якщо прогнозується високе навантаження на нього.

Де знайдено:

* protocol-records
* hr-supervise-all
* group-access-queue
* account-verify

Рекомендації:

* Відмовитись від використання бізнес-ключів де бізнес ключем є повне імʼя ініціатора процесу, оскільки в цих процессах немає взаємодії між декількома користувачами.
* Мінімізувати кількість задач, що виконуються в процесі виконання бізнес-процесу. Наприклад якщо виконується 2 скрипт таски підряд - можна їх обʼєднати в одну. Також розглянути можливість заміни скриптових задач на Execution Listener. Також рекомендується мінімізувати кількість сталих змінних (наприклад initAccToken, або userOrg)

## BP-19. Інше

### BP-19-01. Бізнес-процеси розробників

Критичність: висока

Категорія: безпека

Бізнес-процеси розробників (developer-tools, developer-restore-users) повинні бути виключені з промислового оточення

# Аудит моделі даних

## DM-01. Індекси для критеріїв пошуку

Критичність: висока

Категорія: продуктивність

Відсутність необхідних індексів у базі даних.

Основні наслідки:

* Значне зниження продуктивності запитів, особливо на великих обсягах даних
* Збільшення часу відгуку системи
* Надмірне навантаження на сервер бази даних
* Неефективне використання ресурсів системи

Рекомендації:

* Створити індекси на всі зовнішні ключі (FK)
* Створити індекси на інші поля, окрім унікальних та зовнішніх ключів, по яких відбуваються JOIN таблиць, якщо такі є
* Використати опцію indexing=true для критеріїв пошуку для автоматичного створення індексів які відповідають типу пошуку

Якщо прийнято рішення не використовувати автоматичне створення індексів за допомогою опції indexing=true використати наступні шаблони для сворення індексів в залежності від типу пошуку

для полів типу text та varchar і типу пошуку searchType="equal"

`CREATE INDEX IF NOT EXISTS user\_drfo ON users(lower(cast(user\_drfo as varchar)));`

для типу пошуку searchType="contains"

`CREATE INDEX ix\_user\_work\_data ON users USING gin (user\_work\_data gin\_trgm\_ops);`

для типу пошуку searchType="startsWith" та поля типу text

`CREATE INDEX tbl\_col\_idx ON tbl(col text\_pattern\_ops);`

для типу пошуку searchType="startsWith" та поля типу varchar

`CREATE INDEX tbl\_col\_idx ON tbl(col varchar\_pattern\_ops);`

для типу пошуку searchType="startsWith" та поля типу char

`CREATE INDEX tbl\_col\_idx ON tbl(col bpchar\_pattern\_ops);`

У всіх інших випадках створються звичайні індекси

## DM-04. Перелік колонок в критеріях пошуку

Критичність: висока

Категорія: продуктивність: безпека

Основні знахідки:

* Для більшості полів, що повертаються в Критеріях Пошуку, задається тип пошуку.
* Реальна бізнес-логіка не передбачає пошуку по більшості цих полів.

Наслідки:

* Потенційне створення зайвих індексів при вмиканні indexing= true
* Або необхідність відмовитись від автоматичного створення індексів

Рекомендації:

* Провести детальний аналіз усіх Критеріїв Пошуку та відповідної бізнес-логіки
* Видалити типи пошуку для полів, які не використовуються в реальних пошукових операціях
* Або відмовитись від використання автоматичного створення індексів зазначеного в п. DM-01 і створювати всі необхідні індекси вручну, враховуючи тип пошуку та типи полів

## DM-05. Ліміти на критерії пошуку

Критичність: висока

Категорія: продуктивність, безпека

Описано у секції DM-09.

## DM-06. Нормалізація схеми бази даних

Критичність: висока

Категорія: надійність

### DM-06-1 Відсутність унікальних ключів для полів таблиць, які повинні мати унікальні значення.

Основні наслідки:

* Проблеми з цілісністю даних
* Зниження продуктивності запитів

Приклади:

* team\_case\_statuses.team\_case\_statuses\_numb
* account.teams\_code
* можливо є ще натуральні ключі в інших довідниках

Рекомендації:

* Визначити поля, які потребують унікальних ключів
* Створити відповідні обмеження унікальності в базі даних
* Перевірити наявні дані на дублікати

### DM-06-2 Повна відсутність зовнішніх ключів (Foreign Keys, FK) у базі даних.

Основні наслідки:

* Порушення цілісності даних між таблицями
* Ризик появи "осиротілих" записів
* Складність у підтримці зв'язків між таблицями
* Ускладнення процесу аналізу та розуміння структури бази даних

Приклади:

* doc.case\_status
* time.timeslots\_teams\_code

Рекомендації:

* Провести аналіз зв'язків між таблицями
* Визначити всі необхідні зовнішні ключі
* Створити відповідні обмеження FK в базі даних
* Перевірити існуючі дані на відповідність новим обмеженням

### DM-06-03 Використання JSON-полів для зберігання структурованих даних

Основні наслідки:

* Складність у виконанні запитів та індексації даних, що зберігаються в JSON-полях.
* Потенційні проблеми з продуктивністю при роботі з великими обсягами даних.
* Ускладнення процесу валідації даних на рівні бази даних.
* Можливі проблеми з цілісністю даних.
* Складність у проведенні аналізу та створенні звітів на основі цих даних.

Приклади:

* У таблиці users поле user\_work\_data зберігає додаткову інформацію (edrpou, job, spec, roles) у JSON форматі.
* У таблиці time поле timeslots\_data зберігає складну структуру даних про часові слоти у JSON форматі.

Рекомендації:

* Провести аналіз структури даних, що зберігаються в JSON-полях.
* Розглянути можливість нормалізації даних:
	+ Для user\_work\_data створити окрему таблицю з відповідними полями.
	+ Для timeslots\_data створити окрему таблицю для зберігання часових слотів з відповідними зв'язками.
* Якщо повна нормалізація неможлива
	+ Створити відповідні індекси (Див. DM-01 для типу пошуку searchType="contains")
	+ Впровадити механізми валідації даних на рівні додатку.

Хоча використання JSON-полів може бути виправданим у деяких випадках, рекомендується переглянути поточну структуру даних для покращення продуктивності, забезпечення цілісності даних та спрощення роботи з ними в майбутньому.

## DM-09. Пагінація (Посторінкова навігація) на критеріях пошуку

Критичність: висока

Категорія: продуктивність

Велика кількість запитів що повертають всередньому більше 100 рядків. Деякі запити до 11 000 рядків

Основні наслідки:

* Високе навантаження на сервер БД
* Збільшення часу виконання запитів
* Перевантаження мережі
* Велика кількість даних, що отримується, може перевантажити оперативну пам'ять додатка REST API.
* При збільшенні кількості користувачів або даних навантаження зростатиме нелінійно, що може призвести до збоїв або значного зниження швидкодії.

Приклади:

get\_countries\_v

get\_directory\_account\_spec\_v where account\_spec\_active = $1

get\_org\_v

get\_org\_v where active = $1

get\_org\_v where is\_assigned = $1

get\_org\_v where (is\_assigned = $1 and active = $2)

get\_dir\_mk\_v where active = $1

get\_users\_v where user\_work\_data ilike ($2)

get\_users\_v where user\_roles ilike ($2)

get\_dir\_dr\_v where active = $1

get\_dir\_dr\_dt\_v where active = $1

get\_org\_choice\_v

Рекомендації:

* Передивитись бізнес логіку, чи дійсно там потрібні всі дані. За можливості додати додаткові умови які зменшать кількість рядків що повертаються або використати пагінацію.
* Додати примусовий ліміт на рівні дата-моделі для критеріїв пошуку

## DM-09. Інше

### DM-09-01 Критерій пошуку, що дістає всі дрфо користувачів

Критичність: висока

Категорія: безпека

Вплив:

Будь який автентифікований користувач може вивантажити дрфо всії користувачів в реєстрі

Рекомендації:

Видалити зайві критерії пошуку

# Аудит UI-форм бізнес-процесів

## FR-01. Великі форми

Критичність: висока

Категорія: підтримуваність

Вплив:

* Погіршення користувацького досвіду при заповненні або перегляді форми
* Збільшення ймовірність виникнення системної помилки у користувача при роботі з формою (наприклад, втрата заповнених даних через закінчення сесії користувача)
* Ускладнення тестування форми. Велика кількість тест кейсів, які треба покрити

Приклади:

|  |  |
| --- | --- |
| Назва форми | Кількість компонентів |
| new-order-summary-step-vi | 240 |
| snippet-alert | 236 |
| new-order-summary-v2-final-edit | 219 |
| questionnaire-view | 134 |

Рекомендації:

* Форми для заповнення великої кількості даних варто поділити на групи відповідно до сутностей які заповнюються. Наприклад для форми **details-page-final**, можна розділити персональні дані, контактні данні (включно з довіреною особою), дані про мед заклади на окремі форми
* Для форми **team-control-main,** хоч вона і не є великою по кількості компонентів можна об'єднати дублюючі елементи “Поточна дата” та “Вибір нової дати” шляхом передзаповнення форми редагування
* Для великих форм виключенням може бути підписуючі задачі, де користувачу треба показати інформацію для підписання

## FR-02. Lazy load для компонента Select

Критичність: висока

Категорія: продуктивність

Вплив:

* Додаткове навантаження на сервери та мережу
* Збільшення часу рендерінгу форми
* В деяких випадках при значній кількості компонентів Select у користувача може виникати системна помилка з оновленням авторизаційного токена, так як запити виконуються паралельно

Приклади:

* **team-control-main** – 11 не lazy селектів, що відповідає мінімум 11 одночасним запитам при заході на сторінку форми
* **new-order-summary-v2-final-auth** – 6 не lazy селектів

Рекомендації:

Загалом, в реєстрі активно використовуються select компоненти з lazy load. Але в наведених в прикладах формах ні. Це може бути критично для реєстрів з високим навантаженням, та для “популярних” форм. Для самоперевірки, варто переглядати вкладку Network в DevTools браузера на предмет великої кількості запитів при заході на форму. Також менші форми дозволять легше знаходити подібні випадки

## FR-03. Javascript логіка в компонентах форми

Критичність: середня

Категорія: підтримуваність

Вплив:

* Цей код не перевіряється утилітами для статичного аналізу, аналізу на вразливості, на нього не можливо написати unit тести, по можливості варто відмовитись від його використання
* Велика кількість javascript логіки в компонентах, дуже часто може призводити до помилок. Код в цих полях має тільки базову перевірку на синтаксис, тому часто можуть виникати помилки які важку відловити
* Код виконується від імені користувача який запускає БП, тому помилки можуть призводити до вразливостей безпеки
* Виконання коду, який не відноситься до поля, в контексті якого він викликається, наприклад: навігація, стилізація в полях customValidation та інших, взагалі може призводити до критичних помилок відповідного порталу. Так як розробники платформи можуть вносити зміни як в HTML елементи, так і в адреси сторінок

Приклади:

* Майже у всіх формах в заголовку використовуються додаткові кнопки навігації з кодом
var pathSegments = urlObject.pathname.split('/');
var newPathSegments = pathSegments.slice(0, -2);
newPathSegments.push('process-list');
var newURL = urlObject.origin + newPathSegments.join('/');
window.location.href = newURL;
URL “process-list” може бути змінено в нових версія платформи, або в хотфікс релізах. Також така навігація може зламати функціонал браузера по переходу вперед та назад. Варто розглянути можливість відмови від додаткової навігації на сторінках, або використовувати button з action Navigation
* Також, майже у всіх формах в компоненті columns який використовується для відображення додаткового хідера форми в полі Advanced Conditions. Є велика кількість логіки повʼязаної з навігацією, стилізацією умовним відображенням тексту. Це поле не повинно мати подібну логіку, його призначення - умовне відображення компонента columns. Вся написана там логіка може перестати працювати при змінах в коді платформи
* В компонентах Email використовується customValidation “if (input.includes('.ru'))...”, наприклад в формі **new-order-summary-v2-step-iv**
Платформа має власний, гнучкий функціонал заборонених email адрес, налаштовується через конфігмапу environment-js відповідного порталу в полі emailBlacklist
* adjust-schedule-auth - функціонал друку форми через кнопку “Друк”
Дуже велика кількість JS логіки для кнопки.
Роздрукувати форму можна використовуючи витяги. Робити такі маніпуляції з DOM особливо для реакт та lowcode не варто

Рекомендації:

* По можливості, відмовитись від додаткової навігації, використовуючи вбудовану навігацію порталів, та Navigation Button
* Відмовитись від кнопки Друк на формах, використовуючи функціонал витягів
* Для Email компонент перейти на використання вбудованого blackList email
* По можливості, не використовувати стилізацію через JS логіку

## FR-06. Експериментальні компоненти

Критичність: середня

Категорія: підтримуваність

В деяких випадках критичність може бути високою, так як деякі експериментальні компоненти які вміщають в себе інші компоненти, наприклад well можуть зламати логіку компонентів які вони містять.

Вплив:

* Складні компоненти такі як EditGrid, File, DataImport можуть працювати некоректно, якщо їх поміщено в компоненти не з списку “Oновлені”, потенційно може не працювати наприклад передзаповнення або скрипти компонентів
* Експериментальні компоненти не протестовані розробниками платформи, тому тестування їх роботи в різних комбінаціях з іншими, додаткова робота для розробників форм, регламенту
* Їх відображення може не відповідати стайлгайду прийнятому для платформи, а також не змінитися від перемикання тем інтерфейсу порталів

Приклади:

* Компонент well використовується майже на всіх формах
* Загалом, в усіх проаналізованих формах, знайдено такі експериментальні компоненти - well, datagrid, content, fieldset

Рекомендації:

* Компонент fieldset, має свого аналога зі списку “Оновлених” компонентів. Варто використати його
* Компонент content також має свого аналога зі списку “Оновлених”, він має дещо менший функціонал але його зовнішній вигляд та функціонал протестовано. Варто розглянути його використання
* Компонент datagrid - варто розглянути можливість відмовитись від нього в бік EditGrid, або Columns

## FR-07. Ліміти на критеріях пошуку для компонента Select

Критичність: висока

Категорія: продуктивність

Вплив:

* Додаткове непотрібне навантаження на реляційну базу даних
* Додаткове непотрібне навантаження на сервіс синхронного управління даними реєстру
* Збільшення часу рендерінгу форми

Приклади:

* snippet-alert – addCommunity, addrSettlement, formAddrCommunity, formAddrSettlement
* new-order-summary-v2-final-edit – addCommunity, addrSettlement, addrCommunity, formAddrSettlement
* new-order-summary-v2-step-iii - addCommunity, caddrSettlement, formAddrCommunity, formAddrSettlement

Рекомендації:

Додати ліміти. Навіть коли треба мати ввесь список значень, краще використати розумне значення ліміту. Це пришвидшить роботу самого елементу Select та прибере зайве навантаження на БД.

# Аудит безпеки

## SC-01. Принцип найменших привілеїв

Критичність: висока

Категорія: безпека

Детально описана у секціях SC-04, SC-05

## SC-02. Мінімізація ролей з розширеними правами

Критичність: середня

Категорія: безпека

Вплив:

* Роль demo зазвичай використовується для демонстраційних або тестових цілей, і може не мати належного рівня безпеки. Це створює ризик несанкціонованого доступу до критично важливих бізнес-процесів.

Рекомендації:

Роль demo, яка має доступ до всіх бізнес-процесів повинна бути видалена з промислового оточення

## SC-03. Принцип розділення обов’язків

Критичність: висока

Категорія: безпека

Детально описана у секціях SC-04, SC-05

## SC-04. RLS (Row Level Security) на моделі даних

Критичність: висока

Категорія: безпека

Вплив:

Користувач системи після автентифікації може викликати критерій пошуку і отримати інформацію (рядки з таблиці), доступ до якої не передбачався.

Опис:

RLS дозволяє обмежити доступ для читання до певних рядків (сутностей) на основі зазначеного атрибута користувача. Всі критерії пошуку слід розглядати як відкриті і додавати належні правила на рівні дата-моделі. Для прикладу критерій пошуку get-users, який приймає дрфо як параметр фільтрації, кожен користувач системи може викликати не передаючи цей параметр чи підставити дрфо іншої людини і отримати інформації, доступ до якої не передбачалось давати. Для таких випадків треба додавати примусову фільтрацію (rls) на основі даних з токену

Приклади використання:

* registry-doc-arrange/ Отримати дані по організації суб'єкта з БД: використання критерію пошуку get-users з передачею в параметри пошуку дрфо користувача - ініціатора процесу

РекомендаціЇ:

Додати rls правило, для автоматичної фільтрації результатів пошуку з даних з токену користувача

## SC-05. RBAC (Role Based Access Control) на моделі даних

Критичність: висока

Категорія: безпека

Вплив:

Користувач системи після автентифікації може викликати критерій пошуку і отримати інформацію (критерій пошуку / таблиця), доступ до якої не передбачався.

Опис:

Для забезпечення безпеки даних та гранулярності доступу до даних використовувати механізм RBAC (Role Based Access Control). RBAC дозволяє обмежити доступ до сутності або певних її атрибутів (колонок таблиці) в залежності від ролі користувача.

Де знайдено:

В регламенті не налаштована жодне правило RBAC контролю доступу

Рекомендації:

Створити матрицю контролю доступу для всіх ролей в реєстрі до даних таблиць та колонок і після реалізувати відповідні rbac правила на рівні дата-моделі.

Приклад таблиці:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | katottg | users | goods | accounts | system\_pool |
| user | read | deny | read | read | deny |
| leader | read | read | read | read | read |
| manager | read | read | create | create | read |
| secretary | read | read | read | read | read |

## SC-06. Конфіденційні дані

Критичність: висока

Категорія: безпека

Вплив:

* Ризик витоку даних: Збільшується ймовірність витоку конфіденційних даних, якщо база даних буде скомпрометована.
* Збільшення цілей для атак: Збільшена кількість конфіденційних даних у базі робить її більш цінною ціллю для зловмисників

Де знайдено:

* Збереження токена (детально розписано в розділі BP-07. Робота з несталими (transient) змінними)
* registry-doc-arrange / виклик критерію пошуку get-users - запит отримує у відповіді конфідеційну інформацію (ім’я, телефон, електрона пошта), яка логується в bpms і далі в бізнес-процесі не використовується

Рекомендації:

* Визначити, яка інформація в реєстрі є конфіденційною (дрфо, ім’я, телефон, адреса, серія та номер паспорта). Включно з інформацією, яка зберігається в реєстрі і може бути використана, як тимчасова в бізнес-процесах
* Переглянути необхідність збереження такої інформації взагалі
* Видалити будь яке збереження цієї інформації в персистентних змінних бізнес-процесів, як у відкритому вигляді, так і у вигляді токенів користувача
* Зменшити до мінімуму конфіденційну інформацію, що зберігається в атрибутах користувача у Кейклоак