DOI 10.34185/1562-9945-5-160-2025-15
УДК 004.6

М.А. Краснюк, Вік.В. Гнатушенко, Б.І. Мороз, О.Р. Сокол

РОЗРОБКА АВТОМАТИЗОВАНОЇ СИСТЕМИ ЦЕНТРУ АВТОРИЗАЦІЇ

Анотація. У сучасному цифровому світі, де онлайн-сервіси та додатки стають необхідними інструментами в повсякденному житті, питання безпеки та контролю доступу користувачів набувають особливої важливості. Автоматизована система центру авторизації є сервісом що вирішує ці питання. Сервіс забезпечує безпечне та надійне обслуговування клієнтів різних інформаційних систем. Результатом дослідження є створення програмного продукту, послугами якої можуть користуватися як в інфраструктурі сервісів великої компанії, так і WEB-сервісам. Автоматизована система є достатньо швидкою, що дозволяє обслуговувати велику кількість користувачів із різних інформаційних систем одночасно. Використання послуг автоматизованої системи центру авторизації знімає із бізнесу обтяження зі зберігання паролів, авторизації користувачів в інформаційній системі, впровадження безпекових оновлень та полегшує адміністрування.
Ключові слова: клієнт, сервер, авторизація, пароль, хеш, Erlang, API, WEB-сервіс, інфраструктура сервісів підприємства.

Постановка проблеми. У сучасному цифровому світі, де онлайн-сервіси та застосунки стають необхідними інструментами в повсякденному житті, питання безпеки та контролю доступу набувають особливої важливості. Механізм авторизації є одним з ключових факторів, що забезпечують безпечне та надійне використання цих сервісів.

Головним завданням проведеної роботи була розробка автоматизованої системи здатної до великого інформаційного навантаження. Система повинна була мати підсистеми що слугували середовищем в для адміністрування користувачів бізнесом що є клієнтом автоматизованої системи центру авторизації (надалі AuthHub). При цьому вирішувалося актуальне питання зростаючої потреби в надійній і безпечній авторизації користувачів у світі швидкого технологічного розвитку та зростання загроз у кібербезпеці. Забезпечення безпеки авторизації було і залишається важливим завданням для багатьох організацій та підприємств, які надають свої послуги в Інтернеті, та не тільки. Відсутність ефективної системи авторизації може призвести до витоку конфіденційної інформації, порушення прав доступу, збитків репутаційних та фінансових. Тому важливим питанням було під час проєктування отримати працездатну автоматизовану систему центру авторизації, яка може застосовуватися у різних галузях, забезпечуючи надійний та безпечний доступ до інформації для користувачів.

Аналіз останніх досліджень. Одна з ключових задач авторизаційних систем — це забезпечення безпеки інформації, що передбачає захист від несанкціонованого доступу, маніпуляцій з даними та атак типу DDoS (Distributed Denial of Service). У цьому контексті використання сучасних технологій, таких як хмарні інфраструктури, високопродуктивні бази даних і парадигма акторної моделі, стає важливим підходом до вирішення завдань авторизації [3-5]. Авторизація — це процес надання або обмеження доступу до певних ресурсів або функцій системи на основі визначених прав та ролей користувача. Цей механізм відповідає за перевірку того, чи має користувач дозвіл виконувати певні дії або отримувати доступ до конкретної інформації. Авторизація є ключовим етапом у забезпеченні безпеки інформаційних систем, оскільки вона регулює доступ до конфіденційних даних, критичних операцій та обмежує потенційні загрози, пов’язані з несанкціонованим доступом. Метою авторизації є контроль доступу, захист даних, ефективність розподілу прав, безперервності бізнес-процесів [6,7].

Контроль доступу гарантує, що лише уповноважені користувачі отримують доступ до певних ресурсів. Захист даних забезпечує уникнення витоку чи неправильного використання інформації. Ефективний розподіл прав дозволяє гнучко налаштовувати ролі для різних категорій користувачів. Захист від несанкціонованих змін або втручання, що можуть вплинути на роботу системи забезпечує безперервність бізнес-процесів [8,9].

Центр авторизації є ключовим елементом сучасних інформаційних систем, який забезпечує контроль доступу до ресурсів на основі політик безпеки. Його роль полягає в забезпеченні безпеки даних і ефективності роботи системи, особливо у високонавантажених середовищах. Він має суттєве значення для ефективності, і зменшує навантаження на інші компоненти інфраструктури, в наслідок чого централізація функцій авторизації звільняє інші елементи системи від необхідності виконувати додаткові перевірки доступу, що підвищує їх продуктивність. Центр авторизації також повинен забезпечувати швидкість обробки запитів. Оптимізовані алгоритми та механізми кешування в центрах авторизації повинні дозволяти обробляти запити користувачів у реальному часі навіть під значним інформаційним навантаженням. Сучасні центри авторизації повинні легко інтегруватися з хмарними платформами, що дозволить масштабувати їхню роботу залежно від зростання кількості користувачів чи ресурсів [10,11].

Метою роботи є створення автоматизованої системи центру авторизації, що дозволить бізнес-підприємствам не обтяжувати себе авторизацією користувачів в інформаційній системі за рахунок забезпечення надійності та зручність процесу моніторингу. Програмний продукт повинен бути стійкий до уражень, мати зрозумілий API із функціоналом адміністрування користувачів певної інформаційної системи.

Основна частина. Для досягнення мети дослідження в межах проєкту auth_hub була обрана система керування базами даних (СКБД) PostgreSQL для зберігання та управління даними. PostgreSQL є потужною об'єктно-реляційною системою керування базами даних (СКБД), яка надає широкі можливості для зберігання, організації та оптимізації даних. Вона пропонує розширений набір функцій, що включають підтримку

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

SQL запитів, транзакційності, індексування, виконання збережених процедур, реплікацію даних та багато іншого. Обґрунтування вибору PostgreSQL складається з таких властивостей як надійність, масштабність, розширені можливості, відкритість та широке використання.

Надійність PostgreSQL підтверджується тим, що вона володіє механізмами забезпечення цілісності даних, відновлення після збоїв та захисту від втрати даних. Це важливо для системи, яка обробляє авторизацію користувачів та зберігає чутливу інформацію. Масштабованість PostgreSQL підтверджена можливістю працювати з великими обсягами даних та високими навантаженнями. Вона підтримує горизонтальне та вертикальне масштабування, що дозволяє розширювати систему під потреби росту обсягів даних та кількості користувачів. Розширені можливості PostgreSQL підтверджується багатим набором функцій та розширень, що дозволяють реалізовувати різноманітні завдання. Вона підтримує розширення SQL, географічні дані, розподілену обробку, повнотекстовий пошук та інші розширені можливості, які можуть бути корисними для реалізації системи авторизації користувачів. PostgreSQL є відкритим програмним забезпеченням з активною спільнотою розробників та користувачів. Це означає наявність безлічі документації, підтримки, патчів та оновлень, що забезпечує стабільність та розвиток системи в майбутньому. Широке використання обумовлено тим, що PostgreSQL є однією з найпопулярніших вільних СКБД, вона використовується багатьма великими компаніями та проєктами. Це означає наявність експертів, готових допомогти з розробкою та підтримкою системи.

Структура розробленої системи складається з підсистем, а кожна підсистема має свої ролі. Коренева підсистема auth_hub має простори, тому роль у підсистемі auth_hub надаваться користувачеві саме на простір, а простором є перелік підсистем на які діють ролі підсистеми auth_hub. У користувача “admin” за замовченням є ролі адміністратора у всіх просторах підсистеми auth_hub. Простір був створений для того, щоб надавати ролі на конкретний простір, а не на всі разом. Щоб користувач отримавши роль не мав змоги маніпуляцій або перегляду даних, що не є в його просторі підсистем. Як висновок можна сказати, що коренева підсистема auth_hub має в собі ролі для керування доступом користувачів у сервісі auth_hub, тоді коли ролі в інших підсистемах, є інформацією що експортується до інших сервісів клієнтів. Були встановлені наступні обмеження системи:

- неможливість видалення підсистеми authHub, навіть при наявності на це ролі;
- неможливість видалення користувача admin, навіть при наявності на це ролі;
- неможливість видалення ролі адміністратора “am” із підсистеми “authHub”, навіть при наявності на це ролі;
- неможливість забирати роль “am” у користувача admin у підсистемі “authHub”, просторі “authHub”, навіть при наявності на це ролі.

На кожен функціонал є своя роль у підсистемі authHub. Перелік усіх ролей підсистеми authHub та їх опис зазначено у таблиці 1 (на стан версії сервісу 3.0.0). У таблиці 2 наведено функціонал та перелік ролей, одну з яких необхідно мати для використання функціоналу (на стан версії сервісу 3.0.0).

Таблиця 1

Ролі підсистеми authHub та їх опис

Роль	Опис
am	AdMin - роль адміністратора. Усі права.
la	Local Admin - роль локального адміністратора. Не має права на видалення та створення підсистем.
si	See Info - роль на перегляд інформації підсистеми: можливі ролі, ролі користувачів у цій підсистемі.
cu	Create Users - роль, що дає право на створення користувачів у системі.
du	Delete Users - роль, що дає право на видалення користувачів із системи.
ar	Add Roles - роль, що дає право на додавання ролей користувачам системи.
rr	Remove Roles - роль, що дає право на вилучення ролей у користувачів системи.
cr	Create Roles - роль, що дає право на створення нових ролей у підсистемі.
dr	Delete Roles - роль, що дає право на видалення ролей у підсистемі.
gr	Get Roles - роль, що дає право на отримання ролей користувача в підсистемі.

Під час авторизації користувача в системі створена сесія на 30 хв, завдяки якій користувач може отримувати доступ до сервісів, що є клієнтами автоматизованої системи центру авторизації auth_hub. Сервіси клієнти мають свої технічні логіни та паролі, завдяки яким вони взаємодіють з auth_hub для ідентифікації користувача та отримання його в необхідній підсистемі ролей. У системі auth_hub реалізовано таймер що видаляє вже не актуальні сесії.

Таблиця 2

Функціонал та ролі для взаємодії із підсистемами

Функціонал API	Ролі
Перегляд інформації про ролі користувачів в підсистемі	am, la, si
Перегляд інформації про ролі в підсистемі	am, la, si
Видалення ролей із підсистеми	am, la, dr
Створення нових ролей в підсистемі	am, la, cr
Видалення підсистеми	am
Створення нової підсистеми	am
Створення нового користувача в системі	am, la, cu
Видалення користувача із системи	am, la, du
Додавання ролей користувачу в підсистемі	am, la, ar
Вилучення ролей у користувача в підсистемі	am, la, rr
Створення сесії	_
Перевірка актуальності сесії	_
Отримання логіну по сесії користувача	_
Отримання ролей сесії користувача	am, la, gr

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

У даній роботі ми використовували такі архітектури взаємодії: клієнт-сервер та сервер-сервер. Вони є широко використаними моделями в розробці програмного забезпечення, включаючи серверні проєкти та WEB-системи.

У клієнт-серверній архітектурі, клієнти є користувачами або додатками, які взаємодіють з системою та надсилають запити до сервера. Сервери, зі свого боку, обробляють ці запити та забезпечують відповіді клієнтам. Ця модель передбачає розділення обов'язків між клієнтською та серверною стороною, що дозволяє досягти більшої модульності, масштабованості та керованості системи.

Архітектура проєкту побудована на базі акторної моделі, акторна модель є концепцією паралельного програмування, яка використовується в мові програмування Erlang та багатьох інших мовах, і є ключовим аспектом фреймворка OTP. Акторна модель базується на ідеї процесів (акторів), які взаємодіють один з одним шляхом обміну повідомленнями. Кожен актор має свій внутрішній стан і може виконувати певні дії у відповідь на отримані повідомлення. Актори можуть створювати нових акторів, надсилати їм повідомлення і відповідати на отримані повідомлення.

В розробці було визначено ефективним використовувати для супервайзера стратегію one_for_one, бо в архітектурі проєкту, воркери супервайзеру не мають тісної співпраці між собою. Стратегія one_for_one – це коли один з акторів видає аварійну помилку, супервайзер перезапускає цей актор

Важливими аспектами архітектури системи є сторонні сервіси та бібліотеки. Так було обрано Cowboy, який є невеликим, швидким, сучасним HTTP-сервером для Erlang/OTP. Cowboy прагне забезпечити повний стек HTTP в невеликій базі коду. Його оптимізовано для низької затримки та низького використання пам'яті, частково тому, що він використовує двійкові рядки. Також Cowboy надає можливості маршрутизації, вибірково надсилаючи запити обробникам, написаним мовою Erlang. Оскільки для керування з’єднаннями використовується Ranch, Cowboy можна легко вбудувати в будь-яку іншу програму. Cowboy є важливою частиною сервісу, бо завдяки йому реалізується взаємодія із клієнтами та користувачами. Він має в собі пул воркерів які будемо програмувати, та саме завдяки цьому пулу розроблена автоматизована система зможе одночасно обробляти декілька запитів одночасно.Poolboy - це легка загальна бібліотека пулу для Erlang, яка зосереджена на простоті, продуктивності та надійному аварійному відновленні. Він знадобиться у створені та адмініструванні пулом акторів, що будуть відповідати за взаємодію із базою даних. Пулу було надано назву - pg_pool. Важливим є врегулювання кількості акторів у пулах cowboy та pg_pool, щоб мінімізувати вузькі місця в архітектурі. Вузьке місце - частина програми, або архітектури що затримує швидкодію усієї системи та навантажувальне тестування необхідно саме для виявлення вузьких місць у системі.

Також під час виконання роботи проведено дослідження алгоритмів хешування паролів. В результаті дослідження прийшли висновку, що будемо використовувати в проєкті для хешування паролів користувачів алгоритм SHA-256 з уповільнювачем PBKDF2.

Розробка структури таблиць бази даних була ключовим етапом у створенні інформаційної системи, від правильної організації таблиць залежить ефективність, масштабованість і продуктивність системи. Кожна таблиця матиме свій Primary Key і завдяки цьому було реалізовано зв'язки між таблицями. В результаті проведених досліджень було визначено, що в схемі треба реалізовати 5 функцій, які дозволяють зменшити обсяг запитів до бази даних, що підвищить швидкодію усієї системи:

1. create_subsystem(subsystem, description) - створення підсистеми;
2. delete_allow_role(subsystem, role) - видалення ролі із підсистеми;
3. delete_subsystem(subsystem) - видалення підсистеми;
4. delete_user(login) - видалення клієнта системи.

Реалізація методу. Дослідивши різні типи архітектурного стилю API (Application Programming Interface), було зроблено висновок, що для серверної частини автоматизованої системи центу авторизації краще застосовувати архітектурний стиль REST (Representational State Transfer). Причиною обрання REST є те, що центри авторизації зазвичай мають багато клієнтів (вебдодатки, мобільні додатки, сервіси), які взаємодіють із сервером, також REST забезпечує легку інтеграцію для різних клієнтів, і не вимагає складного налаштування та підходить для малих і середніх команд розробників. Також при сумісному його використанні з текстовим форматом JSON (JavaScript Object Notation), забезпечує швидку передачу даних без перевантаження. Крім того, REST це ефективний вибір для реалізації центру авторизації завдяки його простоті, підтримці сучасних стандартів аутентифікації, низьким затратам на впровадження та обслуговування.

Встановлено, що у рамках серверної частини автоматизованої системи центру авторизації, вхідні дані мають бути отримані від користувачів системи та від бази даних при запиті користувача. Взаємодія із користувачами реалізована завдяки архітектурі REST API. Форматом передаваних даних обрана JSON. Таким чином, API сервіс складається із двох компонентів: клієнтське API та адміністративне API.

Для розробки даного продукту було використано ноутбук компанії Dell, із процесором Intel Core i7, відеокартою Intel(R) Xe Graphics (TGL GT2) та з ОЗУ у розмірі 25 Гб, накопичувач SSD обсягом 1 Тб. Вказані технічні засоби не мають бути обов'язково ідентичних характеристик для комфортної розробки системи. Також для розробки системи встановлено і використано такі програмні засоби:

- операційна система Linux Ubuntu 22.04.2 LTS;
- мова Erlang 25.0.4 Ubuntu jammy;
- компілятор Rebar3 з github репозиторію;
- утиліти операційної системи Linux make та git;
- СКБД postgresql;
- інструмент адміністрування баз даних DBeaver;
- платформа для тестування API Postman;
- IDE IntelliJ IDEA.

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

Для роботи серверної частини даного продукту було використано сервер EC2 орендований у компанії Amazon Web Services (далі AWS), із процесором Intel Core i7, відеокартою Intel(R) Xe Graphics (TGL GT2) та з ОЗУ у розмірі 25 Гб, накопичувач SSD обсягом 1 Тб. База даних була розташована на тому ж хості, що і сервіс auth_hub. При збільшенні кількості користувачів база даних може бути розміщена на іншому хості.

Вказані технічні засоби не є обов'язковими характеристиками для безперешкодної роботи системи.

Для роботи системи було встановлено і використано на сервері такі програмні засоби:

- операційна система Linux Ubuntu 22.04.2 LTS;
- мова Erlang 25.0.4 Ubuntu jammy;
- компілятор Rebar3 з github репозиторію;
- утиліти операційної системи Linux make та git;
- СКБД postgresql;

Адміністрація та налаштування сервісу була реалізована через консоль завдяки інструменту операційних систем сімейства UNIX - SSH. Слід зазначити, що SSH (Secure Shell) - це криптографічний протокол для безпечного доступу до мережевих пристроїв та серверів. SSH дозволяє адміністраторам і користувачам безпечно взаємодіяти з віддаленими системами через незахищені мережі, шифруючи весь трафік між клієнтом і сервером.

Висновки. Результатами проведеної роботи є розроблено програмне забезпечення, яке виконує функції формування та контролю авторизації користувачів. Забезпечення надійності та зручність процесу моніторингу авторизації реалізовано в роботі завдяки спроєктованої структурі та архітектурі системи, використанні API сервіс, реалізації функціонала для реєстрації нових користувачів, в тому числі збереження їхніх особистих даних та інформації для авторизації; розробці механізму перевірки автентичності користувачів під час авторизації, використовуючи захищені протоколи та шифрування даних.

Запропоновано ефективну модель збереження та маніпуляції ролей в підсистемах автоматизованої системи центру авторизації на основі механізмів журналювання та моніторингу активності користувачів, що дозволяє виявляти потенційні загрози безпеці.

В подальших дослідженнях необхідно провести аналіз та тестування розробленого сервісу під великим навантаженням, що дозволить збільшити безпечність автоматизованої системи формування та контролю авторизації.

ЛІТЕРАТУРА

1. Kief Morris. "Infrastructure as Code: Managing Servers in the Cloud." O'Reilly Media, 2016, 362 p.

2. Yevgeniy Brikman. "Terraform: Up & Running: Writing Infrastructure as Code." O'Reilly Media, 2019, 460 p.

3. Сучасні системи автоматичного керування технологічними комплексами: навч. посіб. / А. М. Сільвестров, М. Я. Островерхов та ін. – Київ: КПІ ім. Ігоря Сікорського, 2023. 386 с.

4. Обшта А. Ф., Бугаєць В. В. Комплексний алгоритм для системи програмного обслуговування логістики гуманітарних послуг. Комп'ютерні системи та мережі. 2023. Т. 5, № 1. С. 79-88. doi: 10.23939/csn2023.01.079.
5. D. Soldatenko, Vic.Gnatushenko. Study of efficiency of using it-infrastructure-as-a-service for cloud computing / D. Soldatenko, Vic.Gnatushenko // Системні технологій. Регіональний міжвузівський збірник наукових праць. – Випуск 2(139). – Дніпро, 2022. – С. 68-76 DOI 10.34185/1562-9945-2-139-2022-07
6. Lorin Hochstein, Rene Moser. "Ansible: Upand Running: Automating Configuration Management and Deployment the Easy Way." O'Reilly Media, 2017, 478p.
7. Joe Armstrong. "Programming Erlang: Software for a Concurrent World" [Електорнний ресурс] – Режим доступу до ресурсу: http://shop.oreilly.com/product/9781937785536.do
8. Website "Ukrainian Community of Programmers" [Електорнний ресурс] – Режим доступу до ресурсу: https://jobs.dou.ua/salaries/
9. Francesco Cesarini, Simon Thompson, "Erlang Programming" [Електорнний ресурс] – Режим доступу до ресурсу: http://shop.oreilly.com/product/9780596518189.do.
10. Official documentation website Erlang [Електорнний ресурс] – Режим доступу до ресурсу: https://www.erlang.org/doc/index.html/
11. Official documentation website PostgreSQL [Електорнний ресурс] – Режим доступу до ресурсу: https://www.postgresql.org/

REFERENCES

1. Kief Morris. "Infrastructure as Code: Managing Servers in the Cloud." O'Reilly Media, 2016.
2. Yevgeniy Brikman. "Terraform: Up & Running: Writing Infrastructure as Code." O'Reilly Media, 2019.
3. Modern systems of automatic control of technological complexes: tutorial. / A. Silvestrov, M. Ostroverkhov and oth. – Kyiv: KPI named after Igor Sikorsky, 2023. 386 p.
4. Obshta, V. Buhaiets. Cross-platform software system for the logistics of humanitarian services. Computer systems and networks. Vol. 5, № 1, 2023. pp. 79-88. doi: 10.23939/csn2023.01.079.
5. D. Soldatenko, Vic.Gnatushenko. Study of efficiency of using it-infrastructure-as-a-service for cloud computing / D. Soldatenko, Vic.Gnatushenko // Системні технологій. Регіональний міжвузівський збірник наукових праць. – Випуск 2(139). – Дніпро, 2022. – С. 68-76 DOI 10.34185/1562-9945-2-139-2022-07
6. Lorin Hochstein, Rene Moser. "Ansible: Upand Running: Automating Configuration Management and Deployment the Easy Way." O'Reilly Media, 2017.
7. Joe Armstrong. "Programming Erlang: Software for a Concurrent World" [Elektronnyi resurs] – Rezhym dostupu do resursu: http://shop.oreilly.com/product/9781937785536.do
8. Website "Ukrainian Community of Programmers" [Elektronnyi resurs] – Rezhym dostupu do resursu: https://jobs.dou.ua/salaries/
9. Francesco Cesarini, Simon Thompson, "Erlang Programming" [Elektronnyi resurs] – Rezhym dostupu do resursu: http://shop.oreilly.com/product/9780596518189.do

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

10. Official documentation website Erlang [Elektronnyi resurs] – Rezhym dostupu do resursu: https://www.erlang.org/doc/index.html/
11. Official documentation website PostgreSQL [Elektronnyi resurs] – Rezhym dostupu do resursu: https://www.postgresql.org//

Received 16.06.2025.
Accepted 20.06.2025.

Development of an automated authorization center system

Software has been developed that performs the functions of forming and monitoring user authorization. Ensuring the reliability and convenience of the authorization monitoring process is implemented in the work due to the designed structure and architecture of the system, the use of API service, the implementation of functionality for registering new users, including the storage of their personal data and information for authorization; development of a mechanism for verifying the authenticity of users during authorization using secure protocols and data encryption. An effective model of storing and manipulating roles in the subsystems of the automated system of the authorization center based on the mechanisms of logging and monitoring user activity is proposed, which allows identifying potential security threats.

Краснюк Михайло Андрійович – спеціаліст з розробки програмних засобів, акціонерне товариство комерційний банк «ПРИВАТБАНК»
Гнатушенко Вікторія Володимирівна – доктор технічних наук, професор, завідувач кафедри інформаційних технологій і систем Українського державного університету науки і технології.
Мороз Борис Іванович - доктор технічних наук, професор кафедри програмного забезпечення комп'ютерних систем.
Сокол Олександр - аспірант кафедри інформаційних технологій і систем, Український державний університет науки і технологій.

Krasnyuk Mykhaylo - software development specialist, Joint-Stock Company Commercial Bank "PrivatBank".
Hnatushenko Viktoriia – Doctor of engineering's sciences, Professor, Head of Department of Information Technologies and Systems, Ukrainian State University of Science and Technologies.
Moroz Boris – Doctor of engineering's sciences, Professor, Department of Computer Systems and Software, Dnipro University of Technology.
Sokol Oleksiesander - postgraduate of Department of Information Technology and Systems, Ukrainian State University of Science and Technologies.