КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет радіофізики, електроніки та комп'ютерних систем
Кафедра комп'ютерної інженерії

БАЗА ДАНИХ ОБЛІКОВИХ ЗАПИСІВ КОРИСТУВАЧІВ ОРГАНІЗАЦІЇ ІЗ ВИКОРИСТАННЯМ ДЕКІЛЬКОХ LDAP-ПРЕДСТАВЛЕНЬ

Дипломна робота бакалавра
студента 4 року навчання
Спеціальність: 123 «Комп'ютерна інженерія»
Валерія СРІБНОГО

Науковий керівник:
канд. фіз.-мат. наук Юрій БОЙКО,
доцент кафедри комп'ютерної інженерії

Рецензент:
доктор фіз.-мат. наук Євген ІВОХІН,
професор факультету комп'ютерних наук та кібернетики

До захисту допускаю: Завідувач кафедрою
Юрій БОЙКО

Ухвалено на засіданні кафедри ―__‖_____2022 р., протокол № ____

Київ - 2022

РЕФЕРАТ

Обсяг роботи 50 сторінок, 32 ілюстрації, 3 таблиці, 11 джерел посилань. БАЗА ДАНИХ ОБЛІКОВИХ ЗАПИСІВ КОРИСТУВАЧІВ ОРГАНІЗАЦІЇ ІЗ ВИКОРИСТАННЯМ ДЕКІЛЬКОХ LDAP-ПРЕДСТАВЛЕНЬ

Об’єктом роботи є структура бази даних користувачів Київського національного університету імені Тараса Шевченка на базі каталогу LDAP (Lightweight Directory Access Protocol).

Метою роботи є дослідження особливостей бази даних користувачів університету та побудови у відповідності до них каталогу.

Методом дослідження є аналіз актуальної структури бази даних користувачів, визначення її основних особливостей та вимог щодо її реалізації, із подальшим аналізом відповідності запропонованого каталогу до них.

Результати роботи: виконано аналіз основних особливостей організаційної структури університету з використанням наявних відкритих ресурсів у мережі Інтернет. В роботі було створено каталог у відповідності до особливостей структури університету.

Сфера застосувань: отримані під час виконання роботи результати дають можливість впровадження єдиного каталогу в IT-інфраструктуру університету.

Ключові слова: бази даних, реалізація LDAP, каталог, IT-інфраструктура університету, схема LDAP, 389 Directory Server.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

LDAP - Lightweight Directory Access Protocol або легкий протокол доступу до директорії.

OID - Object IDentifier або ідентифікатор об’єкта.

IANA - Internet Assigned Numbers Authority або адміністрація адресного простору інтернет.

ISO - International Organization for Standardization або міжнародна організація зі стандартизації.

КНУ - Київський національний університет імені Тараса Шевченка.

ABNF - Augmented Backus-Naur Form.

ПОСТАНОВКА ЗАДАЧІ

Сформувати вимоги до середовища збереження інформації спираючись на особливості структури університету як організації.

Необхідно розробити каталог, який буде забезпечувати доступ здобувачів освіти та персоналу КНУ до сервісів мережі університету із використанням їх єдиного облікового запису. Каталог має базуватися на реалізації LDAP, яка надає можливість ефективного надання інформації з мінімальним дублюванням.

Реалізувати рішення за вимогами у вигляді LDAP-каталогу. Додатково він має прив’язувати один централізований об’єкт користувача до всіх сервісів університету.

ВСТУП

Оцінка сучасного стану об'єкта дослідження. База даних користувачів організації – це вагомий елемент кожної з існуючих у світі організацій. Існує безліч реалізацій організації даних записів у мережі організації. У випадку даної роботи розглядається структура КНУ, стан якої на даний момент є не дуже ефективним для використання, оскільки вона являє собою змішаний вигляд бази даних, який одночасно використовує реляційні та ієрархічні бази даних, не зважаючи на необхідність для ефективності роботи використання виключно одної реалізації збереження даних користувачів.

Актуальність роботи та підстави для її виконання. Більшість організацій постійно розширюються і збільшують кількість ресурсів і сервісів, які вони надають. Проблема полягає у контролі того, як саме отримувати доступ до нових можливостей організації і у випадку нашого університету це буде викликати багато навантаження на обслуговуючий персонал при нинішньому виді бази даних користувачів.

Мета й завдання роботи. Мета даної роботи являє собою дослідження особливостей бази даних користувачів університету та побудова нової системи збереження даних користувачів із використанням LDAP-каталогу. Для досягнення даної мети було вирішено такі завдання:

- пояснення особливостей структури бази даних користувачів університету;
- розробка схеми каталогу, для подальшого його впровадження;
- побудова каталогу у відповідності до особливостей інфраструктури університету на базі 389 Directory Server.

Об'єкт, методи й засоби дослідження. Об'єктом дослідження є структура бази даних користувачів університету.

Основним методом дослідження є аналіз основних особливостей наявної структури користувачів та визначення відповідності до особливостей запропонованого LDAP-каталогу. Це все відбувалося шляхом дослідження

наявних у відкритому доступі документів стосовно університету та документації обраної реалізації LDAP.

Можливі сфери застосування. Результати даної роботи будуть використані для впровадження єдиного каталогу LDAP в IT-інфраструктуру університету.

Розділ 1. СИСТЕМИ ЗБЕРІГАННЯ ТА АДМІНІСТРУВАННЯ ОБЛІКОВИХ ЗАПИСІВ КОРИСТУВАЧІВ

1.1 Облікові записи користувачів у організації

У сучасному світі прив'язаність певної особи до певної організації здійснюється на підставі формування певного запису про дану особу у певній структурній одиниці організації. Даний запис може існувати у різних видах представлення, таких як: паперовий вигляд, електронний вигляд та усний вигляд. Дані види представлень мають свої правові властивості, наприклад, останній вид прив'язаності особи до організації є незаконним в межах України, а перший та другий види мають обов'язкове, у відповідності до «Законодавства України про трудове право», та додаткове значення відповідно.

В межах даної роботи електронний вид прив'язаності особи до організації буде розглядатися як потужний інструмент для ефективного керування доступом особи до ресурсів організації, до яких вона має певний рівень доступу, із мінімальною кількістю роботи, яку необхідно виконати для забезпечення даної мети. Дана прив'язаність базується на використанні певного типу запису в мережі організації – облікового запису.

Обліковий запис – особливий вид інформації про особу, який частково або повністю описує її обов'язки та права в межах організації. Даний запис формулюється на підставі вже існуючих офіційних паперових записів організації про особу.

Запис організації про її співробітника є обов'язковим до формування, однак перевірка його існування та його актуальність в ситуаціях недовіри до статусу співробітника внутрішніми службами організації, перед безпосереднім допуском його до виконання обов'язків у організації чи використання її ресурсів, може бути проблематичним, у зв'язку з необхідністю звіряти всі отримані дані від особи із даними, які зберігаються в офіційній документації організації.

В той час із використанням облікового запису з'являється можливість його господарем продемонструвати свій доступ до нього і швидко отримати

внутрішньою службою організації інформації щодо можливості допуску особи до ресурсів, що суттєво зменшує кількість необхідної роботи до виконання роботи щодо перевірки прав та обов'язків співробітника.

Даний обліковий запис створюється та зберігається на певній електронній платформі, яка має можливість запису у себе інформації та надання даної інформації у разі необхідності. Дана платформа являє собою електронну базу даних, яких у сучасному світі існує декілька типів, в залежності від поставленими перед ними задачами. Розрізняють такі основні види баз даних, в можливості яких входить бажаний нами функціонал: реляційні та ієрархічні бази даних.

Перший із двох зазначених видів спеціалізується на загальному зберіганню інформації у собі із рівнопотужною можливостями запису та читання інформації. Інформація зберігається у вигляді таблиць між якими існує можливість створення умовних зв'язків, для забезпечення логічного взаємозв'язку між записами у різних таблицях, наприклад, існує дві таблиці «співробітник» (табл. 1.1.1) та «інвентар» (табл. 1.1.2), із умовною залежністю інвентарю від співробітника. Ця умовна залежність дозволяє нам стверджувати що інвентар не може існувати окремо від співробітників, в той час як співробітник може існувати окремо без інвентарю, однак явно прив'язати його до якогось співробітника ми не можемо, тому, для підтримки зв'язку між таблицями, необхідно створювати додаткове поле у таблиці «співробітник», яке містить у собі серійний номер інвентарю, який приписаний за певним співробітником.

Таблиця 1.1.1

Таблиця «Співробітник»

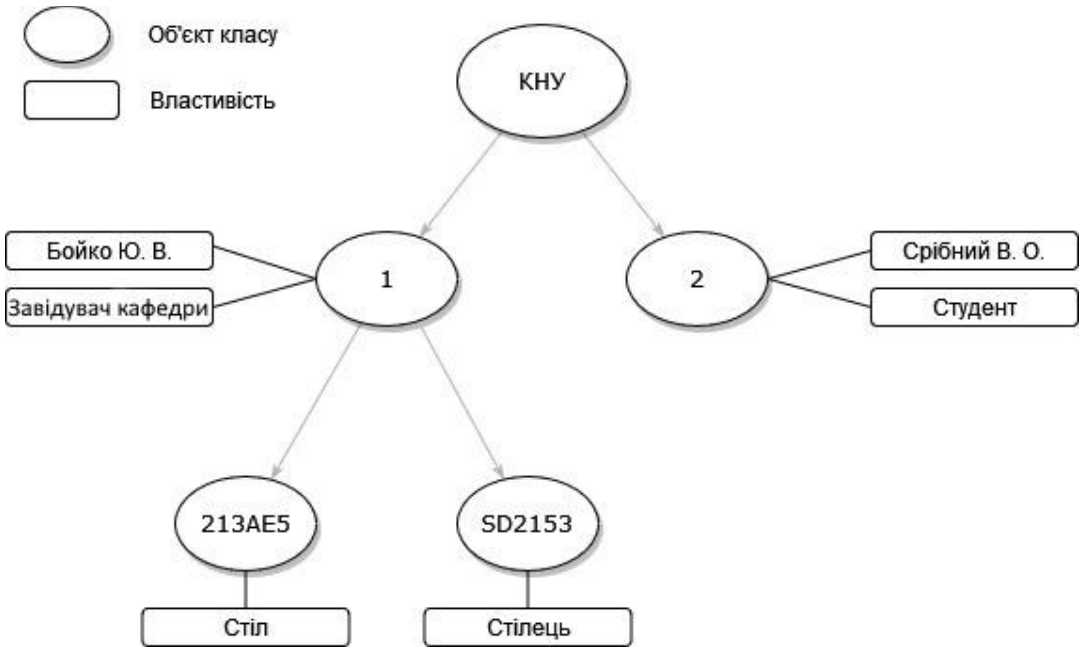
Співробітник			
№	ПІБ	Посада	Інвентар
1	Бойко Ю. В.	Завідувач кафедри	213AE5
1	-	-	SD2153
2	Срібний В. О.	Студент	-

Таблиця 1.1.2

Таблиця «Інвентар»

Інвентар	
Серійний номер	Назва
213AE5	Стіл
SD2153	Стілець

Другий вид баз даних спеціалізується у зберіганні інформації в якості окремих об’єктів каталога, який буде описано у підпункті 1.2, із певними властивостями, які характерні для кожного із об’єктів, із прямою залежністю між об’єктами вигляду клас – підклас. Об’єкт – це аналог рядка у реляційній базі даних, в той час як його властивості – це стовпчики таблиці. Наприклад, у нас є клас «співробітник», у якого є властивості «№», «ПІБ» та «посада», і його підклас «інвентар», із властивостями «серійний номер» та «назва». У порівняні із попереднім прикладом у нас вже є прямий зв’язок між співробітником та інвентарем, що дозволяє нам оминути створення властивості аналогічній полю «інвентар». На діаграмі 1.1.1 зображено приклад вигляду ієрархічної бази даних. Варто зазначити, для охарактеризування об’єкта в даній базі даних, використовується одна з властивостей, яка має бути унікальною.



Діаграма 1.1.1. Приклад ієрархічної подачі даних

Ієрархічні бази даних найкраще показують себе при зчитуванні з них даних, в той час як запис у них може займати більший час по відношенню до реляційних баз даних. Тому у випадках коли база даних використовується в основному для читання і періодичного запису, доцільніше за все використовувати ієрархічний тип.

В даній роботі ми розробляємо базу даних організації для надання доступу до ресурсів організації, що може потребувати дуже багато операцій зчитування з баз даних і рідкого її оновлення, бо ресурси потребують час для їх реалізації. Тому для реалізації завдання роботи ми використовуємо ієрархічну базу даних.

1.2 Каталоги даних організації

«Каталог – це об'єднання відкритих систем, які працюють разом для забезпечення доступу до сервісів каталогу» [1]. В нашому випадку сервісом каталогу є організований набір об'єктів, які є представленням даних організації, до яких необхідно надати доступ, і надалі у роботі даний набір даних буде описуватися як «каталог».

Доступ до каталогу в сучасних комп'ютерних системах забезпечується за допомогою стандартизованого протоколу LDAP, який передбачає собою зв'язок вигляду клієнт-сервер, у якому клієнт надсилає запит на сервер, за яким сервер виконує певні операції в межах каталогу, після чого відправляє відповідь до клієнта. Дані, доступ до яких охарактеризовується даним протоколом, налічують: класи, підкласи та атрибути, які описують кожен із них. Кожен із перелічених видів даних директорії описується у схемі.

Схема – опис компонентів каталогу, який містить характеристики кожного класу та атрибуту [1]. Для класів він може містити безпосередньо опис об'єкта, який представляє даний клас, класи, які по ієрархії стоять вище першого та можливі залежні від нього класи, набір обов'язкових та додаткових атрибутів і тд. Для атрибутів схема містить у собі точний опис атрибута, можливості його багатозначності та інше. Для кожного із об'єктів схеми також задається певний глобальний ідентифікатор об'єкта (OID), який

стандартизується міжнародною організацією стандартизації IANA, для забезпечення однорідності багатьох реалізацій каталогів і уникнення можливих проблем стосовно створення взаємозв'язків між ними. Дані ідентифікатори мають ієрархічну структуру розподілу нумерації, кожна цифра якої охарактеризовує якусь певну структуру, яка описує гілку ідентифікатора. Наприклад, ідентифікатор 1.3.6.1.4.1.8876.2.2 позначає початок ідентифікаторів класів LDAP. 1.3 означає що OID стандартизований за ISO, 1.3.6.1 – ідентифікатор мережі Інтернет, 1.3.6.1.4.1.8876 – OID верхівки willeke.com.[9]

Також для кожної окремо взятої організації допускається створення свого власного зразка схеми, у якому можливе створення своїх унікальних класів та атрибутів, для забезпечення найбільшої ефективності роботи каталогу у мережі організації. Для цього необхідно отримати унікальний ідентифікатор своєї організації, під яким вже створювати набори атрибутів та класів за необхідності.

1.3 Особливості каталогу даних університету

В межах даної роботи проводиться побудова структури каталогу даних для мережі «Київського національного університету імені Тараса Шевченка», задля забезпечення більш ефективного використання ресурсів університету та запобіганню використання багатьох зайвих кроків для реалізації взаємозв'язку між сервісами, які надаються на базі університету.

Наразі в мережі навчального закладу використовується багато різноманітних баз даних, доступ сервісів до яких реалізовано через численні та громіздкі програмні комплекси, що створює зайве та недоцільне навантаження на загальну мережу систем організації університету. Основна база даних відноситься до реляційного виду, що може створювати певні нагромадження вигляду більшого розподілу інформації – і як наслідок більш довгого часу очікування на отримання інформації з бази, за рахунок необхідності витрати більшої кількості ресурсів на складання відповіді від багатьох таблиць.

Перераховані вище проблеми мають вагомий вплив на доступність інформації, що створює значні втрати ефективності роботи, аж до припинення

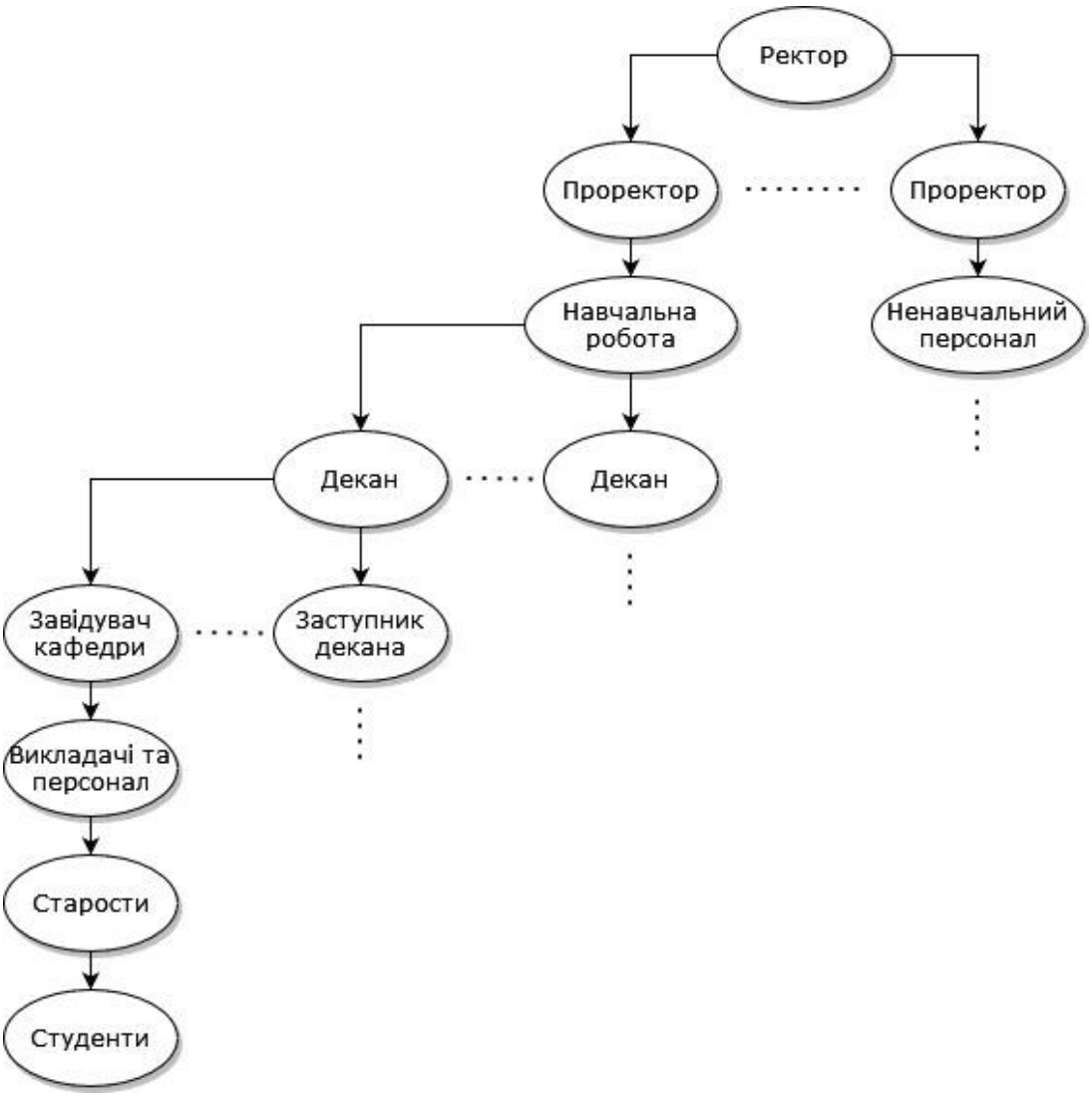
отримання відповідей від бази даних, у пікові моменти навантаження, наприклад одночасне оновлення багатьма підрозділами університету вмісту бази даних. За рахунок цього використання єдиного каталогу для опрацювання даних є необхідністю.

1.3.1 Структура організації

Структура зберігання інформації структурних підрозділів КНУ тягне за собою виконання численних вимог до середовища збереження даних, оскільки дана організація надає доступ до численних ресурсів, якими вона володіє, і доступ до яких необхідно контролювати.

Ці вимоги на сам перед передбачають собою охарактеризування власне структури взаємозв'язків у університеті, які мають ієрархічну будову, в якій знаходиться кожна людина, яка певним чином асоціюється з КНУ. Сама собою дана структура (діаграма 1.3.1.1) має на верхівці ректора, після якого йдуть його проректори і далі по піраміді адміністрування навчальним закладом у плані навчального процесу. Важливо розуміти, що кожна наступна сходинка ієрархії певним чином має зв'язок із попередньою сходинкою, що дає останні право надавати завдання наступній після себе сходинці, в той час як ще вищі сходинки можуть надавати вказівки як другим, так і першим із всіма, хто знаходиться під ними за ієрархією.

Оскільки сама структура університету являє собою ієрархію то доцільно для виконання її адміністрування та керування використовувати ієрархічну структуру збереження даних – каталог, що має безпосередньо збільшити ефективність організації у порівняні з реляційними структурами.



Діаграма 1.3.1.1. Ієрархія університету навчального процесу

1.3.2 Зміна особового складу користувачів

Наступна особливість КНУ полягає у великій кількості користувачів, яким необхідно надавати доступ до ресурсів, які вони маю право використовувати. Загальна статистика університету щодо кількості персоналу навчального закладу передбачає собою, що приблизно 80% всього персоналу – це студенти, які здобувають освіту в університеті. У зв'язку з чим виникає необхідність принаймні двічі на рік активно вносити зміни у базу даних вигляду чи було відраховано студента під час перевірки здобутих знань під час навчального семестру, чи ні. Додатково принаймні раз на рік з'являється необхідність додатково оновлювати дані щодо кількості студентів, у зв'язку з надходженням абітурієнтів на початку навчального року та випуску студентів, які успішно пройшли свої начальні програми та успішно захистили результати

свої досліджень, і отримали відповідний диплом про закінчення навчання та отримання відповідного ступеню.

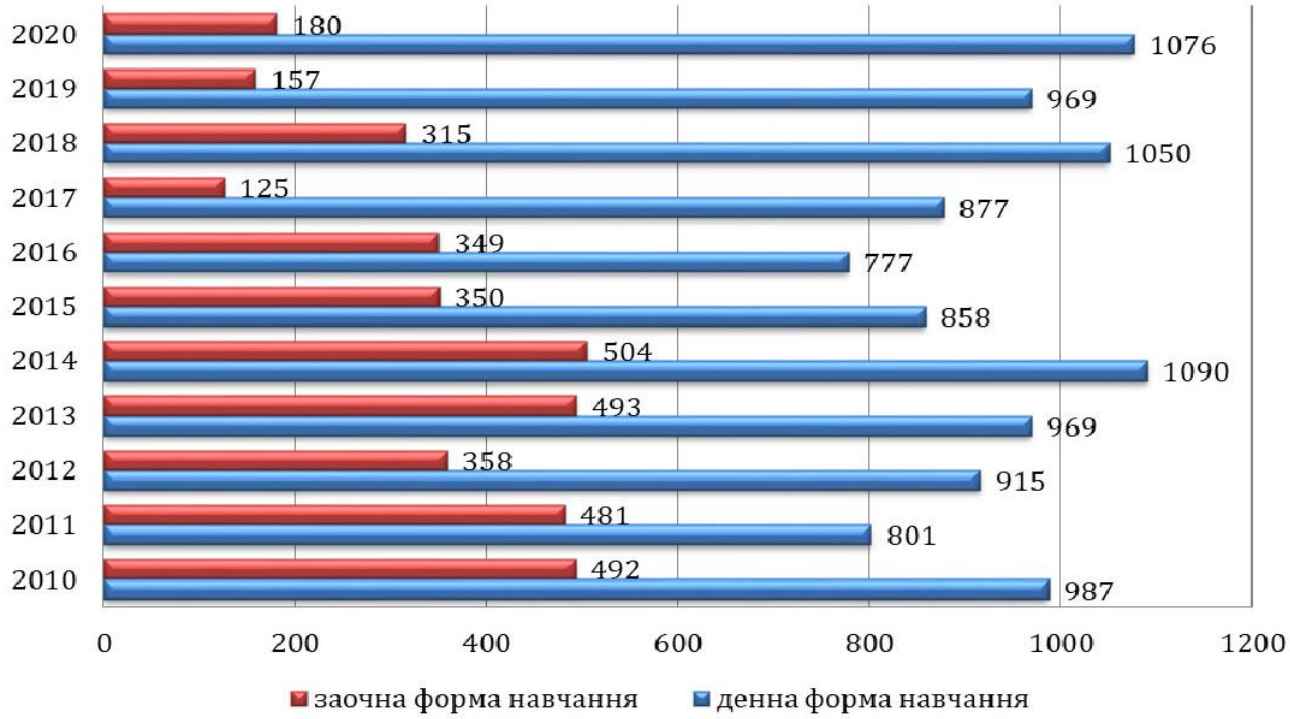
На таблиці 1.3.2.1 зображено статистику, яка була наведена у звіті ректора КНУ за 2020 рік. У даній таблиці зображено загальний контингент студентів на всіх факультетах університету. У відповідності до наведених даних можна побачити тенденцію збільшення загального числа студентів, не зважаючи на те, що студенти також закінчують навчання та виключаються з навчання, за рахунок академічної заборгованості.

Для демонстрації загальної статистики щодо кількості студентів, які не змогли продемонструвати необхідний рівень знань – і були вимушені покинути навчання у КНУ, додатково наведено діаграму 1.3.2.1, яка була взята із звіту ректора університету за 2020 рік.

Таблиця 1.3.2.1

Динаміка зміни сукупного контингенту студентів освітніх та освітньо-кваліфікаційних рівнів бакалавра, спеціаліста та магістра, 2008-2020 рр. [3]

Період	Сукупний контингент		У т. ч.:	
	усього	у т. ч. іногземних громадян	за державним замовленням	за контрактом
2008/09 н.р.	24096	466	14773	9323
2009/10 н.р.	25856	560	15239	10617
2010/11 н.р.	25931	393	15527	10404
2011/12 н.р.	25054	389	15303	9751
2012/13 н.р.	24893	538	15312	9581
2013/14 н.р.	24444	709	14986	9458
2014/15 н.р.	24611	951	15196	9415
2015/16 н.р.	25010	971	15352	9658
2016/17 н.р.	24512	924	15674	8838
2017/18 н.р.	25231	1082	15749	9482
2018/19 н.р.	25934	1291	15387	10547
2019/2020 н.р.	26746	1675	15693	11053
на 01.11.2020	28085	1620	15998	12087

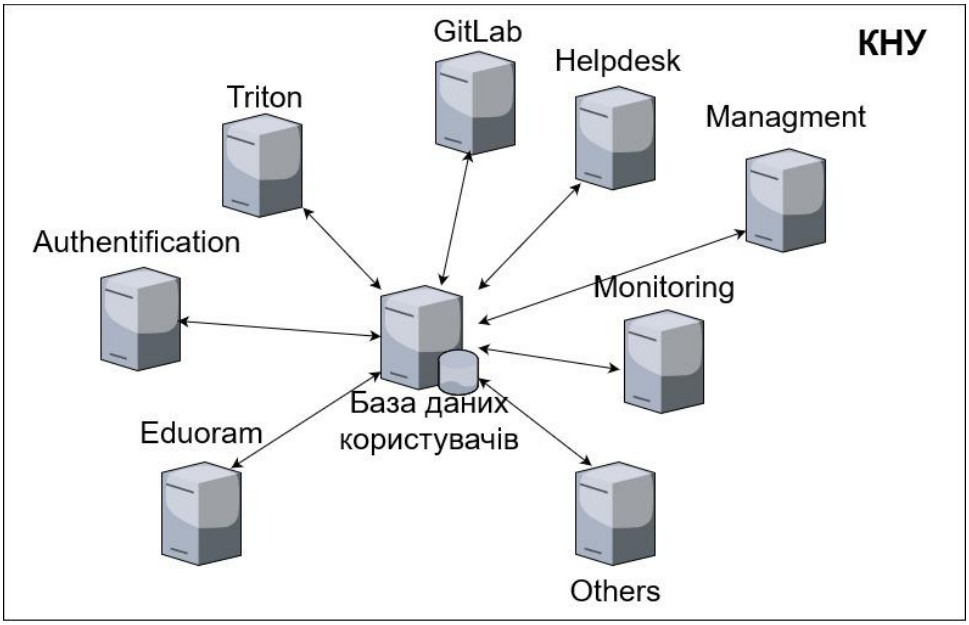


Діаграма 1.3.2.1. Показники відрахування студентів, 2010-2020 рр. [3]

1.3.3 Сукупність ресурсів, які надає університет

У структурі КНУ, кожна особа, яка якимось чином асоціюється з університетом, має доступ до певних численних ресурсів університету. В деяких окремих випадках людина може мати безпосередній доступ до всіх можливих сервісів, які надає університет, як приклад – ректор. У зв'язку з чим на базу даних, у якій зберігаються дані всіх користувачів покладене додаткове завдання вигляду забезпечити ефективний і легкий в реалізації спосіб отримання ресурсами даних про користувачів.

На даний момент часу, в мережі університету існує безліч програмних рішень, які допомагають теперішній базі даних реалізувати даний функціонал. Однак варто зауважити, що сучасна структура даних КНУ є доволі розподіленою і корпоративні дані можуть зберігатися у багатьох місцях одночасно із різними виглядами таблиць, які мають містити одні і ті ж самі дані, тобто таблиці містять подібні за значеннями і описами стовпчики, що призводить до необхідності додаткового ускладнення процесу встановлення рівня доступу користувача до ресурсів. На діаграмі 1.3.3.1 зображено приклад сервісів, які надає університет.



Діаграма 1.3.3.1. Сервіси університету

1.3.4 Представлення даних у відповідності до структури та сервісів університету

Різномаїття ресурсів, які надаються на базі університету, опирається на базу даних. Безпосередньо під кожний ресурс та структурний підрозділ так само має існувати свій окремий запис у базі безпосередньо. З цього запису ми зможемо брати інформацію про користувачів даного ресурсу та їх можливості в його контексті.

Записів про сервіси та структурні підрозділи в каталозі може існувати доволі велика кількість і всі вони використовують одні і ті ж дані користувачів, із невеликою кількістю своїх домішок. Тому розроблена реалізація мусить мати можливість вирішення проблеми постачання однотипної інформації про користувачів для кожного з сервісів із мінімальним дублюванням одних і ти х же даних.

1.4 Програмна реалізація каталогу організації

Для виконання поставленого перед нами завдання необхідно обрати якусь певну реалізацію LDAP-каталогу. Найбільш поширенішими серед них є OpenLDAP, Apache Directory Server, 389 Directory Server, FreeIPA та Active Directory. Принципова різниця між ними полягає лише в реалізації функціоналу

описаного в стандартах LDAP, що робить вибір між ними лише залежним від зручності їх використання.

Безпосередньо для виконання завдання було обрано реалізацію 389 Directory Server. Даний вибір базується на наступних факторах: тип кінцевої операційної системи, на якій розташований сервіс, який буде отримувати дані від каталогу; можливість розширення функціоналу каталогу, який тільки створили та який вже існує тривалий час; методика керування та налаштування каталогу.

Перша особливість пов'язана з тим, що в інфраструктурі університету більшість сервісів, які він надає, розташовані на системах під управлінням Linux. Обрана реалізація безпосередньо також розміщується під управлінням даної операційної системи, що суттєво полегшує можливості її імплементації у вже готову інфраструктуру.

Друга особливість ґрунтується на можливості розширення функціоналу LDAP-каталога в обраній реалізації. Безпосередньо дані можливості впроваджуються через використання вбудованих в реалізацію плагінів, або додаткових елементів управління каталогом. Необхідність даних розширень випливає із важкості реалізації деякого додаткового функціоналу, використовуючи безпосередньо можливості описані в стандартах LDAP-каталогу. До такого виду функціоналу належить використання представлень.

Представлення, або views на англійській мові, відрізняються у реалізації у більшості LDAP-каталогів. В той час як деякі варіанти взагалі не імплементували даної можливості, каталоги, які її мають, по різному її реалізовують. Це пов'язано з фактом відсутності представлень як особливості каталогу, тобто вони відсутні у специфікаціях взагалі. Як такі вони відповідають за фільтрацію даних, які ми хочемо отримати. В контексті обраної реалізації представлення являє собою віртуальне дерево каталогу, яке відображує реальне дерево проте в певному контексті. Даним контекстом є запис у каталозі, який характеризує певну групу, із набору груп, яка потребує дані із іншої гілки каталогу, яка і відображається представленням [11].

Наприклад, у нас є певний набір користувачів, кожен з яких належить певній групі. Належність до даної групи можна реалізувати через безпосереднє вписування користувача у неї із додатковим позначанням користувача членом даної групи або із використанням представлення. Перший спосіб тягне за собою проблему надмірного додавання зайвої інформації, тобто вписування одного користувача в безліч груп і запис до користувача ідентифікаторів цих груп. В той час із використанням представлень можливо лише задати користувачу властивість належності до групи і дістати його у контексті групи. Більш детально про можливості представлень та приклад їх використання буде описано у розділі 3.

Третя особливість є не менш важливою, оскільки завжди є необхідність ефективного та інтуїтивного керування каталогом, чого бракує в деяких реалізаціях. Власне саме управління реалізоване через використання командного рядка, у якому можна вводити команди налаштування каталогу, що є актуальним для більш гнучкого його налаштування, та із використанням веб-інтерфейсу, функціонал якого дозволяє повністю налаштувати каталог із використанням інтуїтивного інтерфейсу. На рисунку 1.4.1 зображено приклад налаштування розширених можливостей серверу каталогу із використанням веб-інтерфейсу.

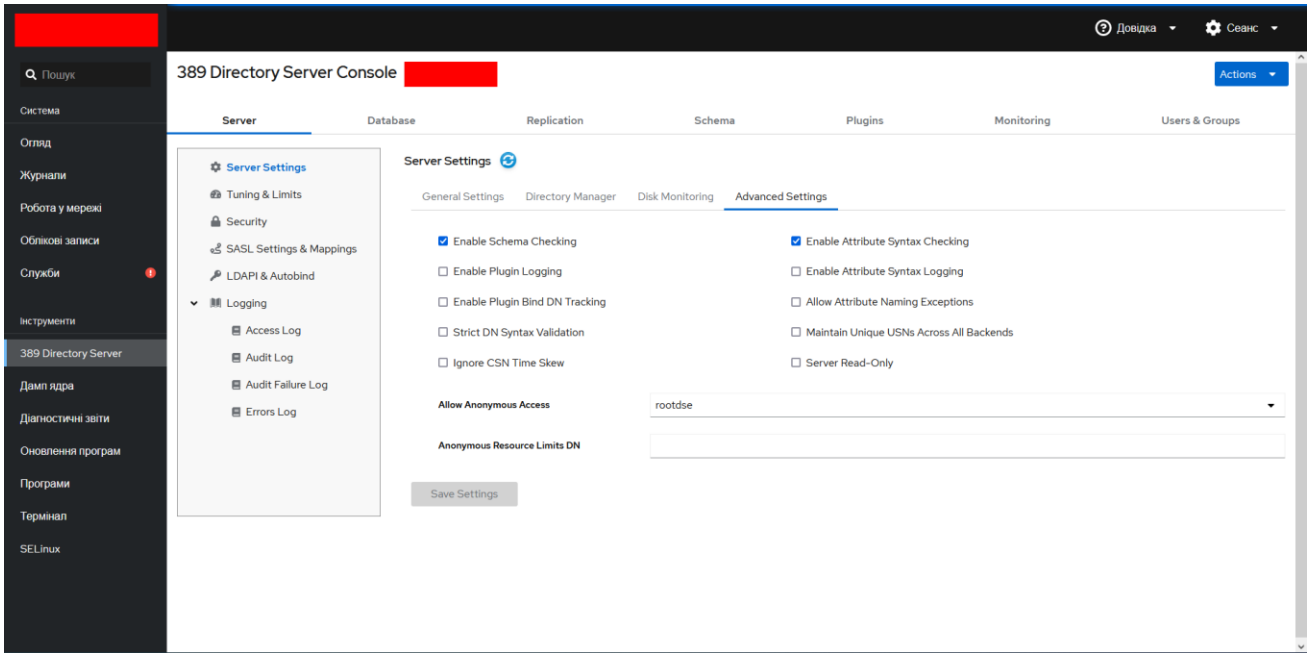


Рисунок 1.4.1. Управління серверу LDAP-каталогу через веб-інтерфейс

Розділ 2. АРХІТЕКТУРА LDAP-КАТАЛОГУ УНІВЕРСИТЕТУ

2.1 Схема як основа каталогу

Схема LDAP-каталогу – це основа каталогу як такого. Безпосередньо даний елемент каталогу відіграє найбільшу роль у його функціонуванні. Як зазначалося у пункті 1.2 першого розділу, вона містить у собі опис елементів каталогу, які складаються з об'єктних класів або Object Classes – елементи які охарактеризовують безпосередньо об'єкт каталогу і які складаються із набору атрибутів, та атрибутів або Attributes – властивостей, що охарактеризовують основну інформацію, що пов'язана з об'єктом, наприклад, ім'я або прізвище користувача. Дані об'єкти частіше за все є стандартизованими і використовуються у відповідності до їх призначення, яке вже може трошки відрізнятися від описаного в стандарті.

Схеми складаються спираючись на синтаксис ABNF [8]. Дане рішення було основане на популярності даного синтаксису на момент опису протоколу LDAP та його каталогів.

Синтаксис для структурних класів має вигляд наведений на рис 2.1.1, в той час як синтаксис для типів атрибутів зображено на рис. 2.1.2. На даних позначеннях присутні умовні позначення характерні для відповідного синтаксису і додаткові позначення описані на рис. 2.1.3.

```
ObjectClassDescription = LPAREN WSP
    numericoid                ; object identifier
    [ SP "NAME" SP qdescrs ]  ; short names (descriptors)
    [ SP "DESC" SP qdstring ] ; description
    [ SP "OBSOLETE" ]         ; not active
    [ SP "SUP" SP oids ]      ; superior object classes
    [ SP kind ]               ; kind of class
    [ SP "MUST" SP oids ]     ; attribute types
    [ SP "MAY" SP oids ]      ; attribute types
    extensions WSP RPAREN

 kind = "ABSTRACT" / "STRUCTURAL" / "AUXILIARY"
```

Рисунок 2.1.1. Синтаксис структурних класів [1]

```
AttributeTypeDescription = LPAREN WSP
    numericoid                    ; object identifier
    [ SP "NAME" SP qdescrs ]      ; short names (descriptors)
    [ SP "DESC" SP qdstring ]     ; description
    [ SP "OBSOLETE" ]             ; not active
    [ SP "SUP" SP oid ]           ; supertype
    [ SP "EQUALITY" SP oid ]      ; equality matching rule
    [ SP "ORDERING" SP oid ]      ; ordering matching rule
    [ SP "SUBSTR" SP oid ]        ; substrings matching rule
    [ SP "SYNTAX" SP noidlen ]    ; value syntax
    [ SP "SINGLE-VALUE" ]         ; single-value
    [ SP "COLLECTIVE" ]           ; collective
    [ SP "NO-USER-MODIFICATION" ] ; not user modifiable
    [ SP "USAGE" SP usage ]       ; usage
    extensions WSP RPAREN         ; extensions

usage = "userApplications"     /  ; user
        "directoryOperation"   /  ; directory operational
        "distributedOperation" /  ; DSA-shared operational
        "dSAOperation"            ; DSA-specific operational
```

Рисунок 2.1.2. Синтаксис типів атрибутів [1]

```
noidlen = numericoid [ LCURLY len RCURLY ]
len = number
oids = oid / ( LPAREN WSP oidlist WSP RPAREN )
oidlist = oid *( WSP DOLLAR WSP oid )
extensions = *( SP xstring SP qdstrings )
xstring = "X" HYPHEN 1*( ALPHA / HYPHEN / USCORE )
qdescrs = qdescr / ( LPAREN WSP qdescrlist WSP RPAREN )
qdescrlist = [ qdescr *( SP qdescr ) ]
qdescr = SQUOTE descr SQUOTE
qdstrings = qdstring / ( LPAREN WSP qdstringlist WSP RPAREN )
qdstringlist = [ qdstring *( SP qdstring ) ]
qdstring = SQUOTE dstring SQUOTE
dstring = 1*( QS / QQ / QUTF8 )   ; escaped UTF-8 string
QQ =  ESC %x32 %x37 ; "\27"
QS =  ESC %x35 ( %x43 / %x63 ) ; "\5C" / "\5c"
; Any UTF-8 encoded Unicode character
; except %x27 ("\'") and %x5C ("\")
QUTF8    = QUTF1 / UTFMB
; Any ASCII character except %x27 ("\'") and %x5C ("\")
QUTF1    = %x00-26 / %x28-5B / %x5D-7F
```

Рисунок 2.1.3. Умовні позначення [1]

Безпосередньо для маніпуляції даними у каталозі необхідно використовувати схеми, оскільки ми лише здатні використовувати ті об'єкти, які у нас є зазначеними у схемах каталогу. Реалізації каталогів можуть містити у собі безліч схем та зазвичай є певний набір стандартних схем, які широко застосовують. Найбільш важливими серед них є схеми core та system.

Схема core – набір класів та атрибутів, обов'язковий для реалізації в будь-якому виді каталогу. Стандартизована і описана у відповідному стандарті

RFC4519. Містить у собі всі необхідні для реалізації каталогу елементи у базовому варіанті, хоча в результуючому каталозі вони навіть можуть і не з’явитися [7].

Схема system – набір класів та атрибутів, які описують основу формування системи каталогу. Елементи даної схеми можна розглядати як обов’язкову верхівку каталогу, якщо буде використовуватися велика кількість об’єктів різного класу, вже від якої ми можемо будувати інші розвилки. Це пов’язано із присутністю в даній схемі атрибутів, які дають можливість охарактеризовувати елементи каталогу як безпосередньо класи [7]. Приклад цього буде наведений у розділі 3.

Реалізація каталогу 389 Directory Server має у собі зазначені вище схеми, що робить можливим базове налаштування каталогу. Дані схеми об’єднані в одну уніфіковану стандартну схему, яка додається під час встановлення і яка містить у собі ще декілька поширених схем.

2.2 Схема каталогу університету

Спираючись на завдання даної роботи ми можемо почати проектувати бажаний нами каталог для зберігання інформації університету. Головною частиною каталогу, як було вже зазначено раніше, виступає його схема, у якій міститься опис об'єктів каталогу, або схеми, що в ньому використовується.

Використовуючи інформацію з пункту 2.1 перед нами постає вибір як саме реалізувати бажану структуру збереження даних: із використанням стандартних схем, які вже реалізовані на серверній частині, чи створити власну схему.

Перший варіант розгортання каталогу є доволі поширеним серед різноманітних організацій, які впроваджують останній у своїй інфраструктурі збереження інформації. Дане рішення базується на достатності характеристик стандартних схем для їй потреб. Як вже зазначалося раніше, стандартні схеми містять у собі мінімально необхідні для повноцінної реалізації каталогу елементи і тому даний варіант є найменш проблематичним для впровадження. Також варто пам’ятати що кожен елемент схеми мусить мати свій власний OID,

отримання якого окремо є доволі важкою задачею в плані затраченого на це часу. Кожен елемент стандартних схем вже має свій власний OID, що прибирає необхідність у їх реєстрації.

Другий варіант розгортання є більш тяжким у впроваджені однак він надає можливість більш гнучкого налаштування необхідних нам властивостей. Особливості реалізації схеми у даному випадку налічують: можливість створення класів та атрибутів, які підходять до типу даних організації без необхідності підшуковувати елементи у вже наявних схемах, маніпулювати вже наявними елементами та окремо стандартизувати структуру своєї організації. Даний варіант розгортання вимагає наявність зареєстрованого організацією OID.

Проектування схеми каталогу університету спиралося на другий варіант розгортання даної структури, для чого на балансі ресурсів університету міститься окремо виділений OID, під реалізацію бажаного варіанту. Даний ідентифікатор має значення – 1.3.6.1.4.1.47428. Варто зазначити що згаданий OID ідентифікує безпосередньо сам КНУ як організацію, тобто ми маємо повну свободу створення бажаних для нас класів та атрибутів, із обмеженням вигляду нумерування ним під ідентифікатором університету, наприклад 1.3.6.1.4.1.47428.1, та подальшого стандартизування новостворених елементів, для більш зручного їх використання у майбутньому.

Безпосередньо при створенні нової схеми для її використанні у каталозі університету, ми маємо спиратися на особливості КНУ, описані у пункті 1.3. На даному етапі проектування не всі особливості беруться до уваги. Основними особливостями, які мають вагомий вплив на структуру схеми, є: структура університету, як організація (пункт 1.3.1), та кількість ресурсів, які він надає (пункт 1.3.3).

Перша особливість вимагає від новоствореної схеми мати можливість створювати об’єкти в каталозі, які зможуть відображати власне структуру університету. Додатково кожен користувач може бути в декількох ланках структури, тобто пов’язаним з університетом більше ніж у одному місці,

наприклад, користувач може бути студентом певного факультету і водночас бути співробітником іншої структури університету, що потребує присутність певної властивості, для пов’язування його.

Друга особливість змушує взяти до уваги факт кількості сервісів, які будуть використовувати інформацію користувача з каталогу. Ця інформація в основному буде одна і та сама, що вимагає передбачення схемою інструментарію для подальшого зв’язування даних між собою. Додатково як і у минулій особливості є ситуації коли користувач є підписаним до більшої кількості сервісів ніж один, що потребує пов’язувального процесу між ними.

Розглядаючи зазначені раніше особливості ми можемо почати формувати об’єкти схеми. Безпосередньо на даний момент часу ми можемо виділити 3 групи об’єктів, які мусять бути описаними у схемі: користувачі, структура університету або окремі структурні групи, до яких належать користувачі, та сервіси, на які може бути підписаний користувач. Зазначені групи ми можемо віднести до об’єктних класів і задати їх власні ідентифікатори OID у відповідності до порядку їх зазначення: 1.3.6.1.4.1.47428.1, 1.3.6.1.4.1.47428.2 та 1.3.6.1.4.1.47428.3. Далі ці групи будуть розглянуті окремо.

2.2.1 Клас користувачів

Розглядаючи користувача як елемент структури університету необхідно виділити його основні особливості. Перш за все кожен користувач мусить мати своє ім’я та прізвище. Ці частини є невід’ємними і мусять бути присутніми у кожного з користувачів. Бажані елементи є вже визначеними та стандартизованими атрибутами LDAP-каталогу і мають відповідні номери OID: name – 2.5.4.41 та sn або surname – 2.5.4.4. Важливо розуміти, що імена та прізвища не є унікальними для людини, оскільки можлива ситуація коли в світі існує дві або більше людини з однаковим прізвищем та іменем, тому вони не можуть однозначно визначати людину у структурі КНУ. Тому необхідно внести наступну властивість – унікальний номерний ідентифікатор, за допомогою якого ми зможемо точно визначити обліковий запис бажаного нам користувача. Для виконання даного завдання ми можемо використати також вже

стандартизований атрибут uid або userid, який ідентифікований за OID 0.9.2342.19200300.100.1.1.

Із зазначенням всіх базових та необхідних особливостей, з'являється необхідність розглянути допоміжні особливості, роль яких пов'язана із забезпеченням сумісності із вже існуючими структурами інформації.

Перше що є необхідним – це визначити атрибут зберігання паролю користувача, що буде охарактеризовувати власність користувачем облікового запису. Додатково даний пароль буде забезпечувати авторизацію при використанні користувачем будь-якого сервісу, оскільки він прив'язаний безпосередньо до першого, а не до останнього, що дозволить користувачу створювати більш складніший пароль для захисту свої даних і запам'ятати лише його, а не створювати багато різних паролів. Властивість userPassword за ідентифікатором 2.5.4.35 підходить для виконання бажаної функції. Також варто зазначити, що даний атрибут зберігає не сам пароль безпосередньо, я його закодований хекс код, що збільшує безпеку його зберігання у каталозі.

Друге що є необхідним – це логін користувача, або унікальне ім'я в каталозі, який буде виконувати аналогічну роль до атрибуту uid. Даний атрибут необхідний для полегшення авторизації користувачів у сервісах, у відповідності до вже існуючого функціоналу бази даних КНУ. Властивість displayName за ідентифікатором 2.16.840.1.113730.3.1.241 має всі необхідні характеристики для забезпечення коректного виконання бажаного завдання.

Перелічені додаткові атрибути не є обов'язковими, як вже було зазначено, але вони полегшують інтеграцію каталогу у вже існуючу інфраструктуру університету. Додатково було передбачено додавання додаткової інформації користувача за допомогою властивості info, OID 0.9.2342.19200300.100.1.4, наприклад номер телефону, електронна адреса і тп.

За допомогою вже визначених властивостей ми можемо створити клас користувача, який зображений на рис. 2.2.1.1.

```
objectClasses: (
  1.3.6.1.4.1.47428.1
  NAME 'Users'
  STRUCTURAL
  MUST ( sn $ name $ uid )
  MAY ( userPassword $ info $ displayName )
  )
```

Рисунок 2.2.1.1. Структурний клас “Users”

Далі вже необхідно додавати властивості у відповідності до вимог структури університету. КНУ – це ієрархічна структура, де користувачі чітко визначені як складові певних структурних підрозділів. Тому з’являється необхідність прив’язування користувача до даних підрозділів. Також варто пам’ятати, що користувач може бути присутнім у декількох гілках структури університету одночасно, наприклад, студент певного факультету і водночас інженер технічного відділу або викладач, який викладає більш ніж на одному факультеті. Дані умови вимагають прив’язування вигляду: підрозділ і відповідна посада у ньому,– що має передбачати можливість атрибуту визначати одне або більше значень одночасно. Атрибута, який би відповідав визначеним умовах, у стандартизованих схемах не існує і тому з’являється необхідність створення свого власного. За рахунок чого було визначено властивість structureInfo, за ідентифікатором 1.3.6.1.4.1.47428.2.1, яка описує бажане прив’язування користувача до підрозділу із його посадою. Додатково даний атрибут використовується для впровадження представлень у кінцевому каталозі, що буде розглянуто у розділі 3. Його визначення зображено на рис. 2.2.1.2

```
attributeTypes: (
  1.3.6.1.4.1.47428.2.1
  NAME 'structureInfo'
  DESC 'Name of ou for nsview filtering and additional info
(position in ou). OU’s in which user is employed and addition
information about his position there.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  USAGE userApplications
  )
```

Рисунок 2.2.1.2. Атрибут “structureInfo”

Додатково для використання даного атрибута було створено окремий структурний клас EmployeePosition, що є підкласом користувача. Це означає що

він не може існувати окремо, а лише як частина користувача. Створення нового окремого класу для використання даного атрибута ґрунтується на можливому розширенні першого у майбутньому, тобто ми не можемо наразі передбачити що конкретно буде використовуватися додатково для пов’язування користувача із структурним підрозділом університету. Даний клас має наступний OID: 1.3.6.1.4.1.47428.1.1. Також для додаткової інформації було додано атрибут info. Власне сам клас зображено на рис. 2.2.1.3.

```
objectClasses: (
  1.3.6.1.4.1.47428.1.1
  NAME 'EmployeePosition'
  SUP Users
  STRUCTURAL
  MAY ( info $ structureInfo )
  )
```

Рисунок 2.2.1.3. Структурний клас “EmployeePosition”

Останнє що необхідно визначити для користувача – властивість для пов’язування його із сервісом, який надає для використання університет. Дана властивість виконує таку ж роль як і атрибут structureInfo, проте вже для сервісів. Оскільки ролі в них майже однакові, то ми вже знаємо що не існує стандартного атрибута, який би зміг виконати бажане завдання, тому було створено властивість serviceInfo, зображено на рис. 2.2.1.4, за ідентифікатором 1.3.6.1.4.1.47428.3.1. Роль даного атрибута полягає у наступному – прив’язування користувача до певного сервісу і його ролі у ньому. Наприклад, користувач користується сервісом моніторингу стану мережі студмістечка і користується ним як звичайний користувач, тобто без всіляких привілеїв. Додатково як і у попередньому випадку для використання даного атрибута нам необхідно створити окремий клас, оскільки попередня проблема актуальна і в даному випадку. Даний клас має ідентифікатор 1.3.6.1.4.1.47428.1.2 і зображений на рис. 2.2.1.5.

```
attributeTypes: (
  1.3.6.1.4.1.47428.3.1
  NAME 'serviceInfo'
  DESC 'Name of a service(s) for nsview filtering and additional
info. Service(s) to what user is subscribed with additional user
based information.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  USAGE userApplications
  )
```

Рисунок 2.2.1.4. Атрибут “serviceInfo”

```
objectClasses: (
  1.3.6.1.4.1.47428.1.2
  NAME 'ServiceSubscription'
  SUP Users
  ABSTRACT
  MAY ( info $ serviceInfo )
  )
```

Рисунок 2.2.1.5. Структурний клас “ServiceSubscription”

Всі створені у даному підпункті властивості мають синтаксис “Directory String”, який описаний за ідентифікатором 1.3.6.1.4.1.1466.115.121.1.15.

2.2.2 Клас груп

Описавши користувача як елемент структури університету, до якої він може відноситися в більш як одному місці, з’являється необхідність описання власне самої структури. КНУ – строго ієрархічна організація, яка складається з багатьох ланок, які взаємодіють між собою. Кожна із цих ланок є унікальною і чітко відповідає за свою сферу діяльності. Додатково на вершині кожної з ланок є певна особа, яка власне неї і керує, що теж необхідно взяти до уваги.

Спираючись на все раніше зазначене ми можемо почати підбір елементів для опису бажаних підрозділів. Перш за все основне що мусить мати структурний підрозділ – це його унікальний ідентифікатор, який буде однозначно його охарактеризовувати у мережі університету. Для виконання даного завдання чудово підходить атрибут gidNumber за ідентифікатором 1.3.6.1.1.1.1.1. Основна задача даного атрибуту, за його описом у схемі, є ідентифікування груп у адміністративному домені середовищі.

Однак ідентифікація лише за номером викликає ускладнення вигляду неможливості одразу дізнатися про який підрозділ йде мова. Найбільше це ускладнює прив’язування користувача до структурного підрозділу, що буде

продемонстровано у розділі 3. Саме тому доцільне додаткове використання ще одного ідентифікатора, з якого вже можна буде чітко визначити до якого підрозділу належить користувач. Даним ідентифікатором може виступати абревіатура підрозділу, наприклад ФРЕКС, або факультет радіофізики, електроніки та комп'ютерних систем, чи ККЕ, або кафедра комп'ютерної інженерії. У випадку важкості ідентифікації підрозділу за абревіатурою, має бути можливість запису повної його назви. Спираючись на дані особливості, було обрано атрибут ou або organizationalUnitName, що охарактеризовує назву структурного підрозділу безпосередньо. Дана властивість ідентифікована OID'ом 2.5.4.11.

Вже описані особливості є важливими в контексті підрозділів університету, однак існують ще певні елементи які необхідно описати. До даних елементів ми можемо віднести керівника підрозділу, та інша додаткова інформація. Для опису керівника ми можемо використовувати атрибут owner, за ідентифікатором 2.5.4.32. Дана властивість не є обов'язковою до визначення оскільки можливі ситуації створення нових структурних підрозділів без керівника, не певний проміжок часу, який відповідає вибору користувача на дану роль. Опис додаткової інформації спирається на необов'язковий атрибут info, який вже був описаний у минулому підпункті.

Результуючий структурний клас, із всіма зазначеними елементами, зображено на рис. 2.2.2.1.

```
objectClasses: (
  1.3.6.1.4.1.47428.2
  NAME 'Group'
  STRUCTURAL
  MUST ( gidNumber $ ou )
  MAY ( owner $ info )
  )
```

Рисунок 2.2.2.1. Структурний клас "Group"

2.2.3 Клас сервісів

Останнє що потребує визначення – це сервіси які надає КНУ. Загалом для опису даних елементів ми можемо керуватися засадами, які були описані у минулому підпункті. До цих засад належать: необхідність однозначно

ідентифікувати сервіс у каталозі, аналогічно до структурних підрозділів в даному випадку доцільно використати цифровий ідентифікатор, додавання додаткового ідентифікатора вигляду абревіатури або скороченої назви сервісу, для полегшення пов'язування користувача з сервісом, визначення того, хто відповідає за сервіс, та додаткова інформація.

Для задовільнення першої засади ми можемо використати атрибут, аналогічний до gidNumber, який було використано для підрозділів університету. Даною властивістю було обрано uidNumber, за ідентифікатором 1.3.6.1.1.1.1.0. Використання аналога, а не безпосередньо атрибута gidNumber, пов'язано із виникненням логічних асоціацій між атрибутами і те що вони описують. Тобто коли ми говоримо про останній, одразу з'являється неявне співставлення його із структурними групами, оскільки безпосередньо gidNumber розшифровується як "group id number", або ідентифікатор групи. При використанні uidNumber вже не можливо привести дане співставлення.

Наступна засада вирішується шляхом використання атрибута name, за ідентифікатором 2.5.4.41, яке в даному контексті визначає ім'я сервісу. Вже описані елементи класу ми відносимо до обов'язкових, аналогічно як і у підпункті 2.2.2. І так само аналогічно до даного підпункту ми додаємо атрибути owner та info для опису відповідального за сервіси і опису додаткової інформації відповідно.

Результуючий структурний клас зображено на рис. 2.2.3.1.

```
objectClasses: (
  1.3.6.1.4.1.47428.3
  NAME 'Service'
  STRUCTURAL
  MUST ( name $ uidNumber )
  MAY ( info $ owner )
  )
```

Рисунок 2.2.3.1. Структурний клас "Service"

2.2.4 Графічне представлення загальної схеми

Описавши всі окремі частини схеми, які будуть використовуватися у побудові каталогу університету, ми можемо об'єднати їх в одну загальну схему.

Безпосередньо сама схема наведена у додатку А, а її графічне представлення наведене на рис. 2.2.4.1.

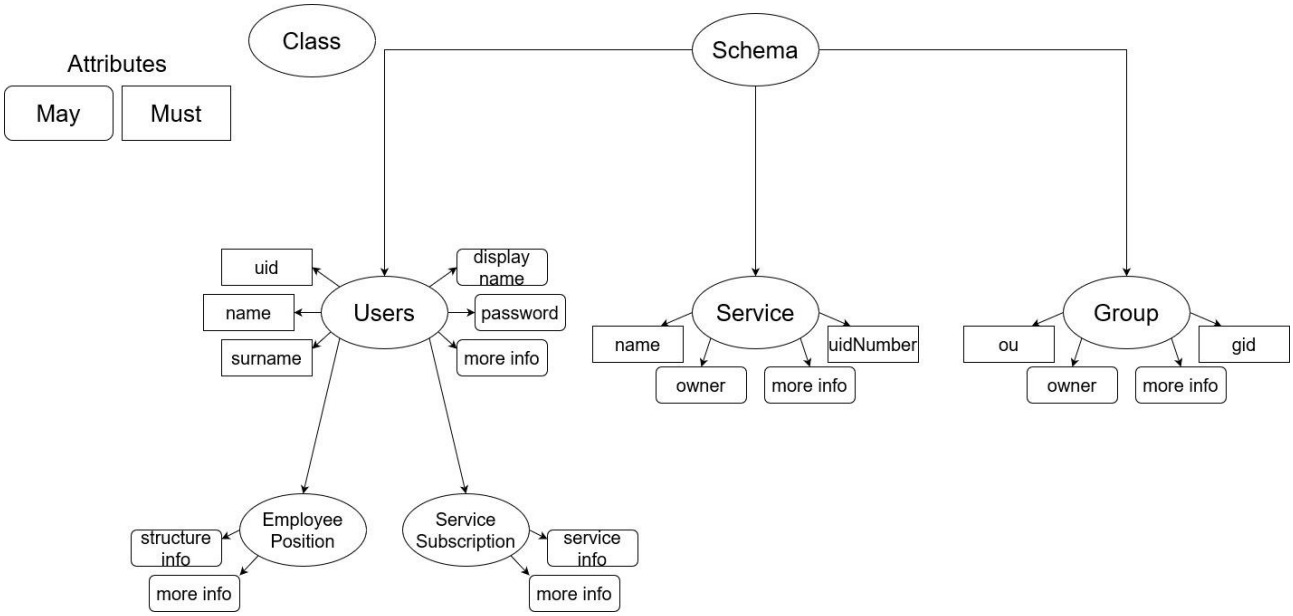


Рисунок 2.2.4.1. Графічне представлення схеми

Загалом рисунок описує все те, що було наведено раніше: структурні класи “Group”, “Service”, “Users” та його підкласи “EmployeePosition” і “ServiceSubscription”, із зазначенням їх відповідних атрибутів і необхідності їх кінцевої присутності.

Додатково кожна безпосередня схема, має бути позначена на початку, що вона відноситься до визначення схеми. Це можна побачити на початку нашої схеми у рядку «dn: cn=schema». Це є необхідність, оскільки всі налаштування LDAP серверу знаходяться безпосередньо у файлах подібного формату до схеми і тому серверу необхідно точно вказати, як сприймати новостворену схему.

Отримана схема відповідає всім поставленим перед нею особливостям і дозволяє побудувати коректний та доступний в розумінні каталог.

Розділ 3. РЕАЛІЗАЦІЯ СТРУКТУРИ LDAP-КАТАЛОГУ

3.1 Каталог як ієрархічна структура

Розробивши та описавши у минулому розділі схему, яка буде покладена в основу LDAP-каталогу, з'являється можливість власне його створення. Варто зазначити, що отримана схема не є чітко визначеною для використання у обраній реалізації, тобто її можливо використати у будь-якій реалізації каталогу, які відповідають відповідним стандартам.

Для реалізації каталогу на базі університету, як вже зазначалося у пункті 1.4 розділу 1, було обрано 389 Directory Server. Дане рішення обґрунтоване можливістю впровадження представлень, як частину каталогу. Більш детально про представлення було сказано в тому ж розділі, що зазначений у минулому реченні.

Перед початком побудови каталогу, необхідно визначити основні його особливості і те, як він буде зберігати дані. Якщо звернути увагу на рис. 2.2.4.1 підпункту 2.2.4 розділу 2, то можна побачити, що сама схема безпосередньо у графічному представлені найбільш доцільно описує структуру каталогу, якою вона мусить бути. Дана структура є наступною: 3 окремі гілки, кожна з яких відповідає за певний елемент загальної інфраструктури університету, а саме користувачів, сервісів та структурних підрозділів. Однак безпосередньо той малюнок є лише графічною абстракцією схеми, яка насправді являє собою перелік необхідних нам класів та їх властивостей, що призводить до необхідності побудувати дійсну діаграму, яка буде відповідати бажаним принципам трьох окремих гілок.

Каталог можна розглядати як дерево, як приклад ієрархічної структури, на якому розташовані різні гілки з інформацією, яка уособлюється в листях гілки. Даний опис чітко описує що таке каталог, у випадку опрацювання даних, і тому можна стверджувати що нам необхідний певний корінь, з якого дане дерево почнеться. У нашому випадку при задані кореню дерева ми мусимо спиратися на певний шаблон, який є задокументованим у реалізації. Даний шаблон дозволяє задати нам ім'я кореневому елементу каталогу. Дане ім'я

власне є властивістю dc або domainComponent, доступний за ідентифікатором 0.9.2342.19200300.100.1.25, яка чітко охарактеризовує корінь. Безпосередньо кореневий елемент зображено на рис. 3.1.1. Назвою, яку він несе, відповідає робочій назві проекту, в основі якого і лежить даний каталог, є knuid.

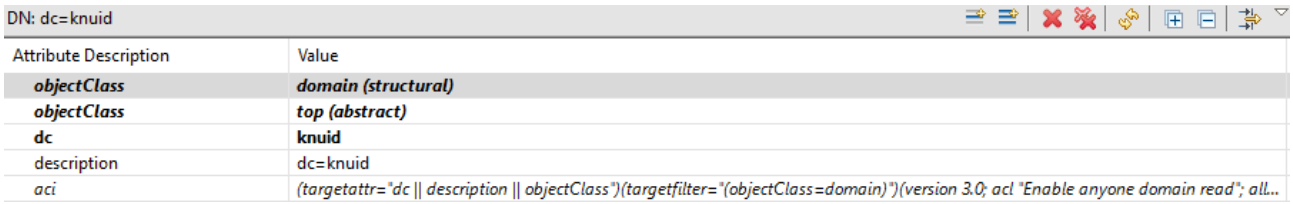


Рисунок 3.1.1. Кореневий елемент каталогу

Від даного кореня і розходяться 3 реалізовані гілки каталогу, які будуть описані далі.

Для демонстрації каталогу та його елементів використовується програмне забезпечення Apache Directory Server, який надає нам змогу маніпулювання даними каталогу, за рахунок чого ми отримуємо дані про об'єкти.

3.1.1 Гілка користувачів

Найголовніша гілка каталогу університету – це гілка користувачів. В даній гілці зберігаються дані про них із використанням розробленої у розділі 2 схеми. В даному випадку основне що ми будемо використовувати це описані у схемі класи "Users", "EmployeePosition" та "ServiceSubscription".

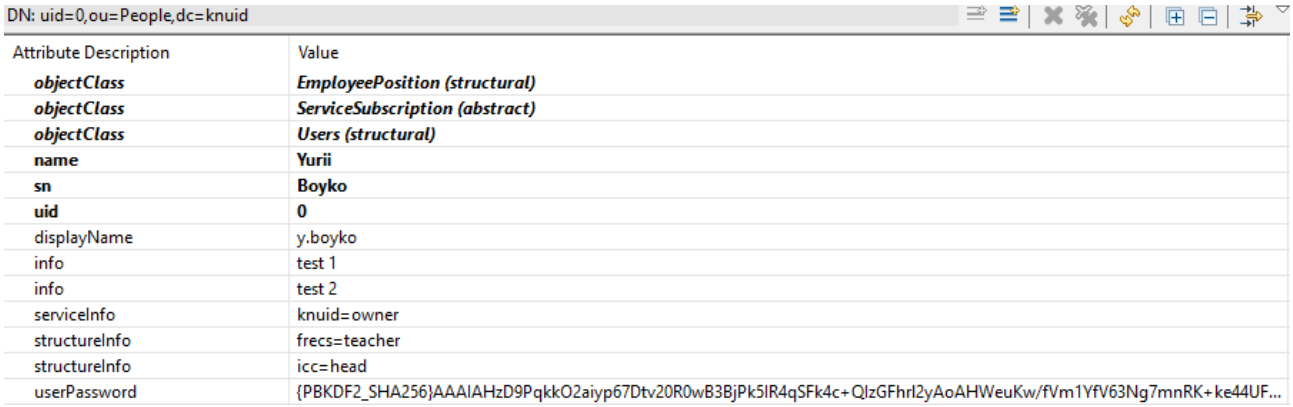
Однак перш ніж заповнювати гілку користувачами необхідно задати її початок. Так само як і з коренем каталогу, нам потрібен «корінь» гілки, для побудови адекватного легкосприйнятного дерева. Для даних потреб ми можемо створити окремий об'єкт, який просто буде вказувати на призначення даної гілки, а саме зберігання облікових записів користувачів. Стандартна практика при реалізації подібного функціоналу включає використання об'єкту класу organizationalUnit із властивістю ou, яка містить значення назви опису об'єктів гілки, що у нашому випадку відповідає "People", яка чітко охарактеризовує даний елемент. Кореневий елемент гілки користувачів зображено на рис. 3.1.1.1. Можна помітити, що у даного об'єкта присутній ще один клас – top. Даний клас є надкласом організаційної одиниці і тому його присутність вимагається стандартом.

DN: ou=People,dc=knuid

Attribute Description	Value
objectClass	***organizationalUnit (structural)***
objectClass	***top (abstract)***
ou	**People**

Рисунок 3.1.1.1. Кореневий елемент гілки користувачів

Із визначеним коренем гілки можна почати створювати об’єкти користувачів. На рис. 3.1.1.2 зображено приклад об’єкту користувача. Даний об’єкт використовує всі властивості свої класів, які були визначені у розділі 2. Певного виділення потребують атрибути structureInfo та serviceInfo, які містять відношення структурний підрозділ і посада в ньому та сервіс і тип користування ним відповідно.



DN: uid=0,ou=People,dc=knuid

Attribute Description	Value
objectClass	***EmployeePosition (structural)***
objectClass	***ServiceSubscription (abstract)***
objectClass	***Users (structural)***
name	**Yurii**
sn	**Boyko**
uid	**0**
displayName	y.boyko
info	test 1
info	test 2
serviceInfo	knuid=owner
structureInfo	frecs=teacher
structureInfo	icc=head
userPassword	{PBKDF2_SHA256}AAAIAHzD9PqkkO2aiyp67Dtv20R0wB3BjPk5IR4qSFk4c+QIzGFhrl2yAoAHWeuKw/fVm1YfV63Ng7mnRK+ke44UF...

Рисунок 3.1.1.2. Об’єкт користувача у гілці

Загалом гілка складається з багатьох таких користувачів. На рис. 3.1.1.3 зображено приклад гілки заповненої трьома користувачами, які в подальшому будуть використовуватися для тестування.

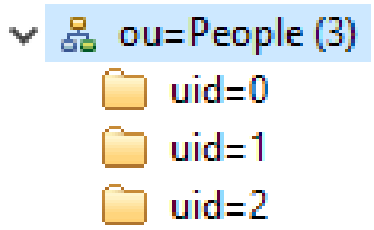


Рисунок 3.1.1.3. Приклад гілки користувачів

3.1.2 Гілка структурних груп

Наступна гілка у каталозі є гілкою структурних підрозділів університету. В дані гілці зберігаються об’єкти, які описують структуру університету. Загалом в даній гілці можна буде спостерігати ієрархічну структуру університету.

Як і гілка користувачів, гілка груп потребує кореня. Загалом структура даного кореня повторює об’єкт “ou=People”, проте в даному випадку атрибут ou приймає значення Groups. На рис. 3.1.2.1 зображено корінь гілки груп. Додатково можна помітити присутність нового ідентифікатора класу у даного об’єкта, а саме nsView. Даний об’єктний клас відповідає за імплементацію представлень у даній реалізації LDAP-каталогу і його присутність буде прокоментована трошки пізніше.

Рисунок 3.1.2.1. Кореневий елемент гілки груп

Визначивши кореневий елемент, можна почати заповнювати безпосередньо заму гілку. Структура університету в певному вигляді є ієрархією організацій, з яких і складається сам КНУ. Наприклад, у нас є безліч факультетів, які створюють власні підгілки, в яких існують кафедри і тп. В даний момент часу повністю структуру університету не буде складено, оскільки даний етап потребує наявності інформації із всіх структурних підрозділів університету і не тільки.

Безпосередньо в даний момент ми можемо скласти наступну структуру: головна надорганізація, тобто те у що всі під організації власне об’єднуються, і певний набір підорганізацій. Приклад даної підгілки зображено на рис. 3.1.2.2. Дана підгілка налічує КНУ, як головну організацію, та «Факультет радіофізики, електроніки та комп’ютерних систем» разом із «Інформаційно-обчислювальний центр» (ІОЦ), як її під організації.

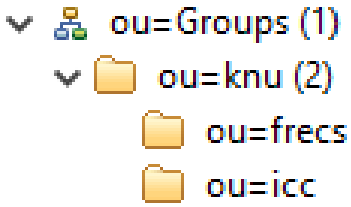


Рисунок 3.1.2.2. Приклад гілки груп

Кожен із елементів зображеної гілки має подібну будову до елемента “ou=icc”, який зображено на рис. 3.1.2.3., тобто складаються із одних і тих самих атрибутів і типів класів. Дані властивості налічують: об’єктні класи Group, який було створено у розділі 2, nsView, який вказує на участі даного об’єкта у формуванні представлень, та top, який підтягується як надклас попереднього класу, і атрибути, характерні для створеного нами класу, gidNumber, ou й owner, які у відповідності до рисунку мають відповідні значення “1”, “icc”, “uid=0,ou=People,dc=knuid”, який вказує на користувача Юрія Володимировича Бойка. Також в даному елементі каталогу налічується властивість nsViewFilter, яка містить у собі значення “(structureInfo=icc*)”. Даний атрибут дозволяє побудувати безпосереднє представлення, яке було описано у пункті 1.4 розділу 1. Більш детально побудову представлень буде описано у пункті 3.2.

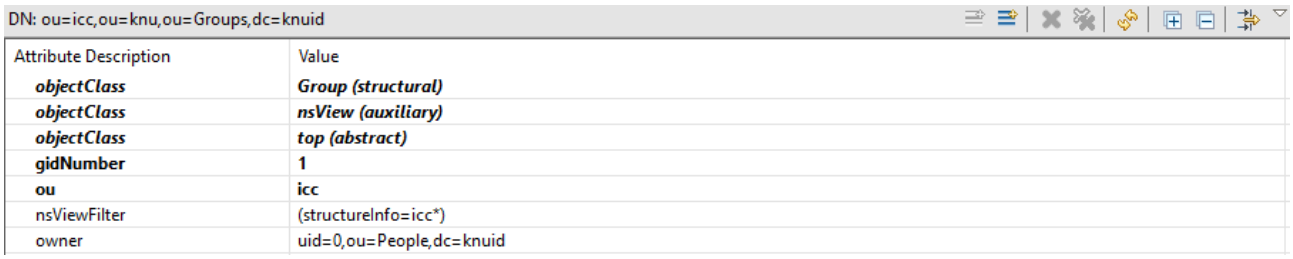


Рисунок 3.1.2.3. Об’єкт групи у гілці

Кінцева гілка груп може налічувати довільну кількість підрозділів університету, у вказаному на рисунку форматі, і повторювати ієрархію їх відносин.

3.1.3 Гілка сервісів

Остання гілка каталогу – гілка сервісів. Як і у минулих гілках, вона має кореневий елемент, який задає її зміст. Даний корінь (рис. 3.1.3.1) повністю повторює собою корінь гілки структурних груп, проте значення атрибуту ou складає “Services”, що в кінцевому випадку ідентифікує даний об’єкт як “ou=Services,dc=knuid”.

DN: ou=Services,dc=knuid

Attribute Description	Value
objectClass	***nsView (auxiliary)***
objectClass	***organizationalUnit (structural)***
objectClass	***top (abstract)***
ou	**Services**

Рисунок 3.1.3.1. Кореневий елемент гілки сервісів

Під коренем в даній гілці знаходять об’єкти, які описують сервіси надавані університетом. Для прикладу наразі дана гілка заповнена лише одним сервісом, для якого і створювався каталог. Даний сервіс має назву «knuid». Безпосередній приклад самої гілки зображено на рис. 3.1.3.2.

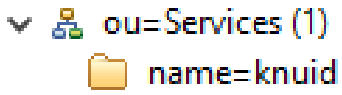


Рисунок 3.1.3.2. Приклад гілки сервісів

Приклад елементу гілки зображено на рис. 3.1.3.3. На даному етапі розробки каталогу, даний елемент дуже подібний до елементу структурного підрозділу, за винятком використання визначеної для неї атрибутики у схемі. Дана проблема пов’язана з тим, що без загальної інформації про всі підрозділи університету та його сервісів ми можемо скласти лише загальну картину.

DN: name=knuid,ou=Services,dc=knuid

Attribute Description	Value
objectClass	***nsView (auxiliary)***
objectClass	***Service (structural)***
objectClass	***top (abstract)***
name	**knuid**
uidNumber	**0**
nsViewFilter	(serviceInfo=knuid*)
owner	uid=0,ou=People,dc=knuid

Рисунок 3.1.3.3. Об’єкт сервісу в гілці

Зображений на рисунку елемент налічує свій відповідний структурний клас, клас nsView, для побудови представлень з користувачами даного сервісу, клас top, як надклас попереднього, та наступні атрибути: name, uidNumber, owner та nsViewFilter. Кількість сервісів для створення в межах даного каталогу є необмежена, за винятком фізичних обмежень вигляду нестачі ресурсів системи, на якій розташований каталог.

3.1.4 Графічне представлення загального каталогу

Описавши кожну з гілок окремо, можна їх скласти в єдиний каталог. Безпосередньо отриманий каталог, який графічно зображено на рис. 3.1.4.1, є лише прикладом, оскільки реальна IT-інфаструктура університету налічує набагато більше інформації, ніж та що була наведена. Тобто в університеті набагато більше ніж 3 користувача, 1-ого сервісу та 3 структурних підрозділів. Безпосередній каталог наведено у додатку Б.

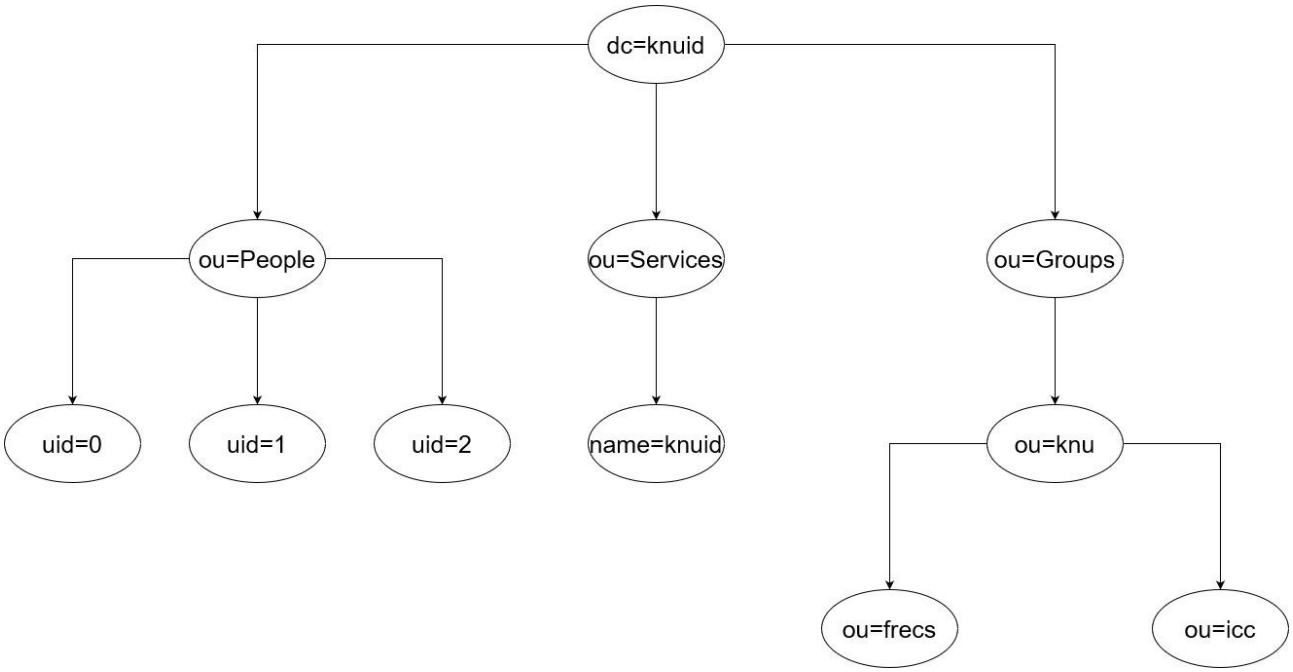


Рисунок 3.1.4.1. Графічне представлення каталогу

3.2 Отримання даних з каталогу

Кожен окремий елемент створеного каталогу адресується із використанням визначного атрибута кожного елемента на шляху від початкового до кореня каталогу. Наприклад, у відповідності до рис. 3.1.4.1 об’єкт нульового користувача, ми можемо отримати за ідентифікатором “uid=0,ou=People,dc=knuid”. Дане адресування відповідає стандарту реалізації LDAP-каталогу та описане у ньому ж.

Всі дані користувача заховані у об’єкті або обліковому записі даного користувача. Дана інформація цікавить нас в контексті структурних груп та сервісів університету. Уявімо що в нас є месенджер, який надається університетом, і кожний користувач зареєстрований в ньому і в той же час у

нас є сервіс електронної пошти, у якому вони також зареєстровані. Два даних різних сервісів використовують одні і ті самі дані користувача, наприклад ім’я, прізвище, пароль входу і тд. Бажана інформація вже є у об’єкті користувача, що в результаті вимагає від сервісу в своєму контексті знайти дану інформацію у каталозі, для її відображення. Стандартна практика пов’язування даних користувача із сервісом полягає у використання атрибутів member та memberOf у сервісі і користувачі відповідно. В даних атрибутах зберігають повні ідентифікатори користувача та сервісу відповідно. Дані ідентифікатори чітко вказують на об’єкт і в контексті сервісу це дозволяє нам вже напряму звернутися до об’єкта користувача за інформацією. Приклад даного використання даних зображено на рис. 3.2.1. Проблему даної практики можна побачити одразу – дублювання однієї і тої ж інформації, тобто ми вказуємо у користувачі його прив’язаність до сервісу і у сервісі вказуємо наявність в ньому користувача. Додатково у користувачі необхідно буде додати властивість, яка буде окремо описувати роль першого у сервісі, що призведе до недоречного ускладнення каталогу.

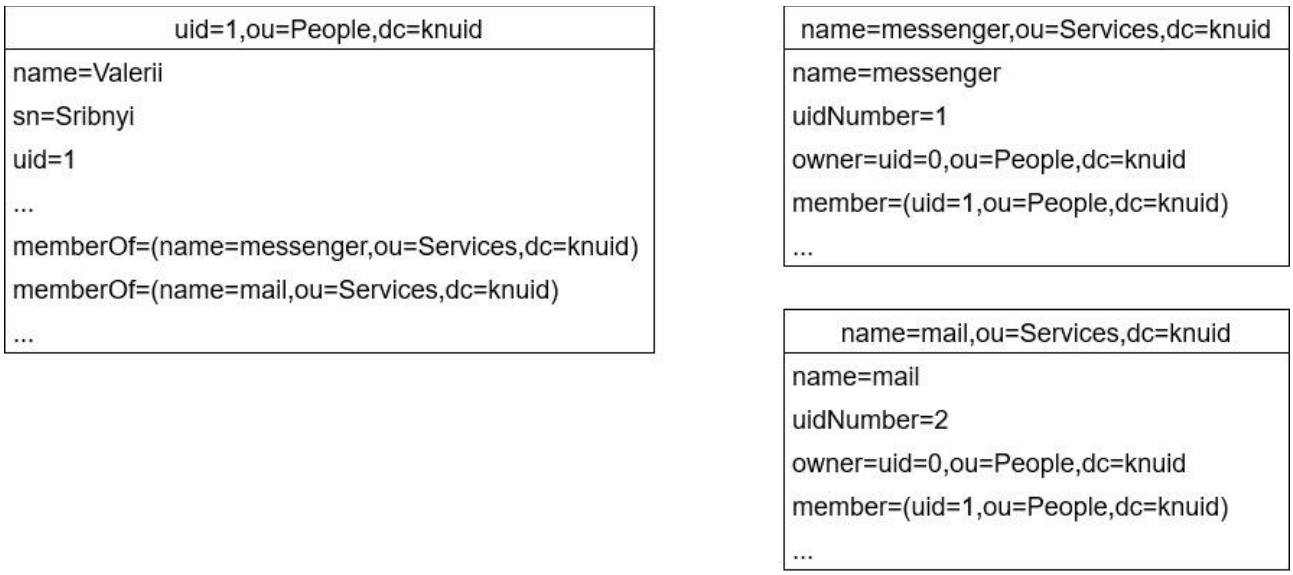


Рисунок 3.2.1. Стандартне пов’язування користувача із сервісами

Із використанням представлень реалізації 389 Directory Server ми отримуємо можливість спростити цей процес, а саме створити певний фільтр у сервісі, який буде відбирати користувачів, які будуть підходити за даним фільтром. Так було зроблено в результуючому каталозі, і якщо ми переробимо

наш приклад за цією стратегією ми отримаємо те, що зображено на рис. 3.2.2. В даному випадку вся інформація про пов’язування користувача із сервісом знаходиться лише у користувача і вже у формі, з якої можна додатково дістати його роль в контексті даного сервісу. Щодо сервісів вони отримують спеціальний фільтр, за допомогою якого вони можуть знайти об’єкти своїх користувачів за рахунок пошуку їх каталогом.

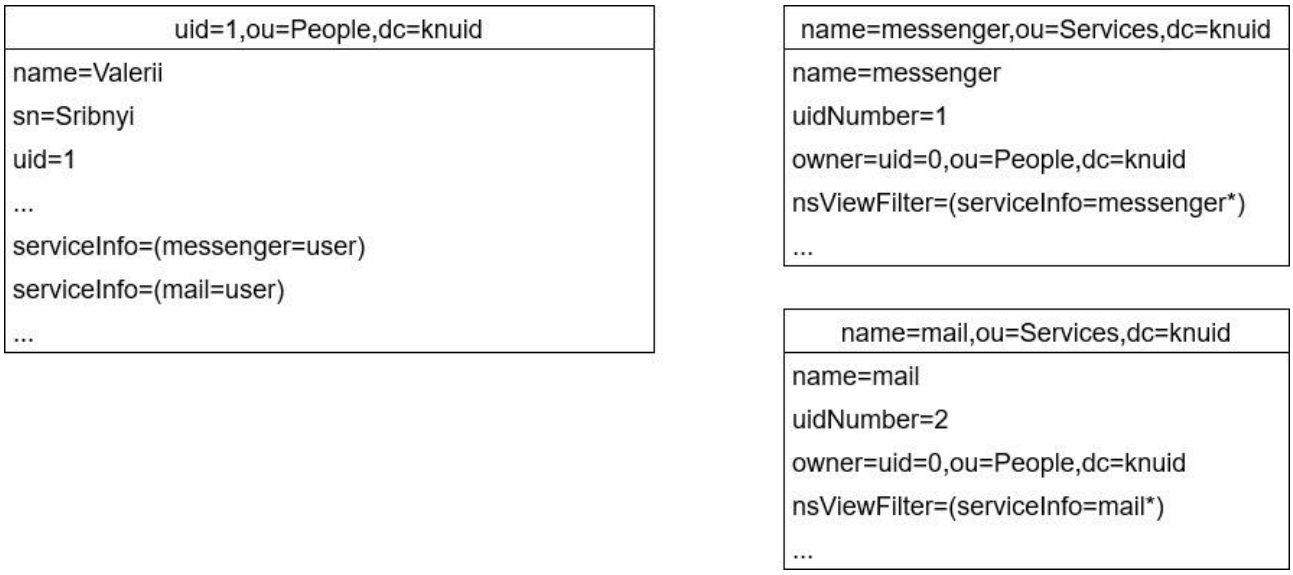


Рисунок 3.2.2. Пов’язування користувача із сервісами із використанням представлень

Безпосередній результат отримання набору користувачів кінцевого каталогу, що є учасниками певної структурної групи, у її контексті зображено на рис. 3.2.3. Безпосередній пошук виконується з контексту підрозділу ІОЦ, що за стандартом LDAP робиться лише на рівні даного об’єкту та нижче. Це означає, щоб отримати користувачів групи, необхідно їх визначити на тому ж або нижчому рівні гілки, що в нашому випадку не так. На рис. 3.1.4.1 ми бачимо що користувачі знаходяться і групи знаходяться у різних гілках, що робить неможливим їх безпосередній пошук за стандартом. Проте вся гілка груп має властивість класу представлень, що дозволяє змістити межу пошуку по гілці до найвищого елемента з класом nsView [2]. Іншими словами кореневий елемент гілки структурних підрозділів та всі елементи під ним мають відповідний клас, що дозволяє у свою чергу вже дійти до безпосередніх

користувачів, кореневий об’єкт яких знаходиться не тому ж рівні, що і відповідний об’єкт груп. Саме тому пошук дав результат [10].

```
ldapsearch -x … -b "ou=icc,ou=knu,ou=Groups,dc=knuid"
"structureInfo=*"
# extended LDIF
#
# LDAPv3
# base <ou=icc,ou=knu,ou=Groups,dc=knuid> with scope subtree
# filter: structureInfo=*
# requesting: ALL
#

# 0, People, knuid
dn: uid=0,ou=People,dc=knuid
info: test 1
info: test 2
displayName: y.boyko
sn: Boyko
name: Yurii
userPassword:: …
serviceInfo: knuid=owner
structureInfo: icc=head
structureInfo: frecs=teacher
uid: 0
objectClass: EmployeePosition
objectClass: Users
objectClass: ServiceSubscription

# 1, People, knuid
dn: uid=1,ou=People,dc=knuid
serviceInfo: knuid=developer
displayName: v.sribnyi
sn: Sribnyi
name: Valerii
structureInfo: icc=ops
structureInfo: frecs=student
uid: 1
objectClass: EmployeePosition
objectClass: Users
objectClass: ServiceSubscription
info: test

# search result
search: 2
result: 0 Success

# numResponses: 3
# numEntries: 2
```

Рисунок 3.2.3. Пошук співробітників ІОЦ

Як видно з результатів пошуку, нам вдалося дістати двох із трьох користувачів нашого каталогу в контексті ІОЦ. Це два користувача є

співробітниками даного підрозділу, відповідно до їх об’єктів. В подальших демонстраціях вивід буде обмежено лише до ідентифікаторів, приналежностей та логінів користувачів.

На рис. 3.2.4 та рис. 3.2.5 зображено результат пошуку користувачів в контексті ФРЕКС та сервісу “knuid”.

```
ldapsearch -x … -b "ou=frecs,ou=knu,ou=Groups,dc=knuid"
"structureInfo=*" dn displayName structureInfo
# extended LDIF
#
# LDAPv3
# base <ou=frecs,ou=knu,ou=Groups,dc=knuid> with scope subtree
# filter: structureInfo=*
# requesting: dn displayName structureInfo
#

# 0, People, knuid
dn: uid=0,ou=People,dc=knuid
displayName: y.boyko
structureInfo: icc=head
structureInfo: frecs=teacher

# 1, People, knuid
dn: uid=1,ou=People,dc=knuid
displayName: v.sribnyi
structureInfo: icc=ops
structureInfo: frecs=student

# 2, People, knuid
dn: uid=2,ou=People,dc=knuid
displayName: t.test
structureInfo: frecs=student

# search result
search: 2
result: 0 Success

# numResponses: 4
# numEntries: 3
```

Рисунок 3.2.4. Пошук представників ФРЕКС

З результатів пошуку членів факультету можна стверджувати, що всі три користувача належать до його структури. В той же час ці три користувача також користуються сервісом “knuid”.

```
ldapsearch -x … -b "name=knuid,ou=Services,dc=knuid"
"serviceInfo=*" dn displayName serviceInfo
# extended LDIF
#
# LDAPv3
# base <name=knuid,ou=Services,dc=knuid> with scope subtree
# filter: serviceInfo=*
# requesting: dn displayName serviceInfo
#

# 0, People, knuid
dn: uid=0,ou=People,dc=knuid
displayName: y.boyko
serviceInfo: knuid=owner

# 1, People, knuid
dn: uid=1,ou=People,dc=knuid
displayName: v.sribnyi
serviceInfo: knuid=developer

# 2, People, knuid
dn: uid=2,ou=People,dc=knuid
displayName: t.test
serviceInfo: knuid=user

# search result
search: 2
result: 0 Success

# numResponses: 4
# numEntries: 3
```

Рисунок 3.2.5. Пошук користувачів knuid

З загальних результатів пошуку можна зробити висновок, що каталог може надавати дані користувачів в контексті підрозділів університету та його сервісів без надмірного їх ідентифікування в об’єктах даних елементів IT-інфраструктури. Додатково використання представлень дозволяє ідентифікувати користувача у будь-якому сервісі або групі за рахунок можливості задання довільного фільтру представлення.

ВИСНОВКИ

Проведений в роботі аналіз показав доцільність впровадження ієрархічного рішення для збереження інформації про користувачів (каталогу LDAP) для нашого університету.

Існуюча IT-інфраструктура університету характеризується надлишковістю та демонструє неузгодженість своєї роботи за рахунок чисельних дублюючих механізмів, які слабо узгоджені між собою.

Розроблене в роботі рішення дозволяє сервісам та групам отримати дані єдиного облікового запису користувача, без їх дублювання у своїй структурі з мінімальною необхідністю їх налаштування для цього.

Схема, як покладена в основу каталогу університету, дозволяє описати вже наявну IT-інфраструктуру університету та у разі необхідності доповнити її додатковими атрибутами.

Розроблений каталог дозволяє масштабувати набір сервісів та структурних підрозділів університету за рахунок можливості довільного налаштування фільтрів представлень для пошуку об’єктів користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Rfc4512, Lightweight Directory Access Protocol (LDAP): Directory Information Models [Електронний ресурс] – Посилання для доступу: https://datatracker.ietf.org/doc/html/rfc4512 (дата звернення 09.09.2021)
2. 389 Directory Server Documentation [Електронний ресурс] – Доступ за посиланням: https://directory.fedoraproject.org/docs/389ds/documentation.html (дата звернення 15.09.2021)
3. Звіт ректора Київського національного університету імені Тараса Шевченка за 2020 рік – Доступ за посиланням: http://www.univ.kiev.ua/pdfs/zvit/zvit-rektora-2020.pdf (дата звернення 01.10.2021)
4. Understanding and Deploying LDAP Directory Services / [Timothy A.. Howes, Tim Howes, Mark Smith, Gordon S. Good] / Addison-Wesley Professional – 2003. – 899 с – ISBN 0672323168.
5. IEEE Internet Computing / Institute of Electrical and Electronics Engineers – 2004. – 66 – 72 с – ISSN 1089-7801
6. LDAP System Administration: Putting Directories to Work / Gerald Carter / "O'Reilly Media, Inc." – 2003. – 312 с – ISBN 0596551916.
7. Rfc4519, Lightweight Directory Access Protocol (LDAP): Schema for User Applications [Електронний ресурс] – Посилання для доступу: https://www.ietf.org/rfc/rfc4519.txt (дата звернення 10.10.2021)
8. Rfc4234, Augmented BNF for Syntax Specifications: ABNF [Електронний ресурс] – Посилання для доступу: https://www.ietf.org/rfc/rfc4234.txt (дата звернення 19.05.2022)
9. LDAP Object Identifier [Електронний ресурс] – Посилання для доступу: https://ldapwiki.com/wiki/OID (дата звернення 25.09.2021)

10. ldapsearch(1) — Linux manual page [Електронний ресурс] – Посилання для доступу: https://man7.org/linux/man-pages/man1/ldapsearch.1.html (дата звернення 25.05.2022)
11. Red Hat Directory Server 11 Administration guide [Електронний ресурс] – Посилання для доступу: https://access.redhat.com/documentation/en-us/red_hat_directory_server/11/html-single/administration_guide/index (дата звернення 07.10.2021)

ДОДАТОК А

```
#
#############################################################################
#
dn: cn=schema
#
#############################################################################
#
attributeTypes: (
  1.3.6.1.4.1.47428.2.1
  NAME 'structureInfo'
  DESC 'Name of ou for nsview filtering and additional info (position in ou). OU's in
which user is employed and addition information about his position there.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  USAGE userApplications
  )
#
#############################################################################
#
attributeTypes: (
  1.3.6.1.4.1.47428.3.1
  NAME 'serviceInfo'
  DESC 'Name of a service(s) for nsview filtering and additional info. Service(s) to
what user is subscribed with additional user based information.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  USAGE userApplications
  )
#
#############################################################################
#
objectClasses: (
  1.3.6.1.4.1.47428.1
  NAME 'Users'
  STRUCTURAL
  MUST ( sn $ name $ uid )
  MAY ( userPassword $ info $ displayName )
  )
#
#############################################################################
#
objectClasses: (
  1.3.6.1.4.1.47428.1.1
  NAME 'EmployeePosition'
  SUP Users
  STRUCTURAL
  MAY ( info $ structureInfo )
  )
#
#############################################################################
#
objectClasses: (
  1.3.6.1.4.1.47428.1.2
  NAME 'ServiceSubscription'
  SUP Users
  ABSTRACT
  MAY ( info $ serviceInfo )
  )
#
#############################################################################
```

```
#
objectClasses: (
  1.3.6.1.4.1.47428.2
  NAME 'Group'
  STRUCTURAL
  MUST ( gidNumber $ ou )
  MAY ( owner $ info )
  )
#
##############################################################################
#
objectClasses: (
  1.3.6.1.4.1.47428.3
  NAME 'Service'
  STRUCTURAL
  MUST ( name $ uidNumber )
  MAY ( info $ owner )
  )
#
##############################################################################
#
```

ДОДАТОК Б

```
# knuid
dn: dc=knuid
description: dc=knuid
dc: knuid
objectClass: top
objectClass: domain

# Groups, knuid
dn: ou=Groups,dc=knuid
objectClass: organizationalUnit
objectClass: top
objectClass: nsView
ou: Groups

# Services, knuid
dn: ou=Services,dc=knuid
objectClass: organizationalUnit
objectClass: top
objectClass: nsView
ou: Services

# People, knuid
dn: ou=People,dc=knuid
objectClass: organizationalUnit
objectClass: top
ou: People

# 0, People, knuid
dn: uid=0,ou=People,dc=knuid
info: test 1
info: test 2
displayName: y.boyko
sn: Boyko
name: Yurii
userPassword:: …
serviceInfo: knuid=owner
structureInfo: icc=head
structureInfo: frecs=teacher
uid: 0
objectClass: EmployeePosition
objectClass: Users
objectClass: ServiceSubscription

# 1, People, knuid
dn: uid=1,ou=People,dc=knuid
serviceInfo: knuid=developer
displayName: v.sribnyi
sn: Sribnyi
name: Valerii
structureInfo: icc=ops
structureInfo: frecs=student
uid: 1
objectClass: EmployeePosition
objectClass: Users
objectClass: ServiceSubscription
info: test

# 2, People, knuid
```

```
dn: uid=2,ou=People,dc=knuid
displayName: t.test
sn: Test
name: Test
structureInfo: frecs=student
serviceInfo: knuid=user
uid: 2
objectClass: EmployeePosition
objectClass: Users
objectClass: ServiceSubscription
info: test

# knu, Groups, knuid
dn: ou=knu,ou=Groups,dc=knuid
ou: knu
gidNumber: 0
objectClass: Group
objectClass: top
objectClass: nsView

# icc, knu, Groups, knuid
dn: ou=icc,ou=knu,ou=Groups,dc=knuid
owner: uid=0,ou=People,dc=knuid
nsViewFilter: (structureInfo=icc*)
ou: icc
gidNumber: 1
objectClass: Group
objectClass: nsView
objectClass: top

# frecs, knu, Groups, knuid
dn: ou=frecs,ou=knu,ou=Groups,dc=knuid
nsViewFilter: (structureInfo=frecs*)
ou: frecs
gidNumber: 2
objectClass: Group
objectClass: nsView
objectClass: top

# knuid, Services, knuid
dn: name=knuid,ou=Services,dc=knuid
nsViewFilter: (serviceInfo=knuid*)
owner: uid=0,ou=People,dc=knuid
uidNumber: 0
name: knuid
objectClass: Service
objectClass: nsView
objectClass: top
```