Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мультимедійних систем факультету інформатики

Інтеграція Smart Data Space з блокчейн-платформою OKP4 для монетизації вуглецевих викидів

Текстова частина до дипломної роботи
за спеціальністю „Прикладна математика” - 113

Керівник курсової роботи
Гороховський К. С.

_____
(підпис)
“____” _____ 2024 р.

Виконав студент
Шелемей Я.О.
“____” _____ 2024 р.

Київ 2024

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мультимедійних систем факультету інформатики

ЗАТВЕРДЖУЮ

Зав.кафедри математики,

проф., д.ф.-м.н.

_____Р. К. Чорней

(підпис)

«___»_____2024 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на кваліфікаційну роботу

студенту Шелемей Ярослав Олеговичу факультету інформатики 4 курсу

Тема Інтеграція Smart Data Space з блокчейн-платформою OKP4 для монетизації вуглецевих викидів

Зміст ТЧ до кваліфікаційної роботи:

Індивідуальне завдання

Анотація

Вступ

Розділ 1: Проблематика вуглецевих викидів

Розділ 2: Огляд технологій Smart Data Space і OKP4

Розділ 3: Інтеграція Smart Data Space з блокчейн-платформою OKP4

Висновки

Список літератури

Дата видачі „27” жовтня 2023 р. Керівник _____

(підпис)

Завдання отримав _____

(підпис)

Тема: Інтеграція Smart Data Space з блокчейн-платформою OKP4 для монетизації вуглецевих викидів

Календарний план виконання роботи:

№ п/п	Назва етапу кваліфікаційної роботи	Термін виконання етапу	Примітка
1.	Отримання завдання на кваліфікаційну роботу.	жовтень 2023	
2.	Обговорення деталей теми на розробка плану виконання	листопад-грудень 2023	
3.	Огляд технічної літератури за темою роботи та опрацювання матеріалів	березень 2024	
4.	Написання теоретичної частини роботи	квітень-травень 2024	
5.	Розробка практичної частини та фіналізація практичної	квітень-травень 2024	
6.	Аналіз отриманих результатів, обговорення остаточної доповіді з керівником	травень 2024	
7.	Здача кваліфікаційної роботи	червень 2024	

Студент: Шелемей Я.О

Керівник: Гороховський К. С.

“27” жовтня 2023

Зміст

Анотація 5
Вступ 6
Розділ 1. Проблематика вуглецевих викидів 7
1.1. Глобальне потепління 7
1.2. Вплив вуглецевих викидів на глобальне потепління 10
1.3. Парниковий ефект 12
1.4. Вуглецеві кредити 15
Розділ 2. Огляд технологій Smart Data Space і OKP4 19
2.1. Smart Data Space 19
2.2. Основи блокчейну 20
2.2.1. Блокчейн 20
2.2.2. Хешування SHA-256 23
2.2.3. Смарт контракти 28
2.3. AXONE (OKP4) 30
2.4. Принцип роботи AXONE 32
2.4.1. Онтології 32
2.4.2. Управління 34
Розділ 3. Інтеграція Smart Data Space з блокчейн-платформою OKP4 37
3.1. Завантаження необхідних даних: 37
3.1.1. Авторизація та отримання токена 37
3.1.2. Виконання HTTPS запитів з використанням отриманого токену 39
3.2. Інтеграція отриманих даних з блокчейн платформою OKP4 42
3.2.1. Підготовка необхідних інструментів 42
3.2.2. Використання мови програмування Rust для написання смарт контракту 49
3.2.3. Додавання даних у блокчейн AXONE 57
Висновки 65
Список літератури 68

Анотація

У даній роботі розглянуто базові принципи та причини глобального потепління, участь у ньому вуглецевих викидів. Проведено роботу з інтеграцією даних про вуглецеві викиди з ферм з блокчейн-платформою **OKP4** для подальшого зберігання та публічного доступу до цієї інформації.

Ключові слова: блокчейн, глобальне потепління, вуглецеві викиди, OKP4, Smart Data Models, Axone, Keyrock, смарт контракт.

Вступ

У сучасному світі проблема глобального потепління є однією з найбільш актуальних та нагальних. Зростання температури планети, спричинене людською діяльністю, відкриває нові виклики та загрози для екосистем, людського здоров'я та економіки. Одним з джерел викидів парникових газів є агропромисловий сектор, зокрема вуглецеві викиди від ферм. Задля мінімізації впливу агротехнологій на зміни клімату було створено системи збору та аналізу даних про викиди пов’язані з діяльністю ферм. Результатом цієї роботи стане ефективне зберігання зібраних даних у блокчейні з можливістю публічного доступу до них.

Актуальність роботи: Глобальне потепління – один з найбільших викликів в історії людства. Зменшення впливу людини на зміни клімату важлива задача для кожного. Завдяки прозорому доступу до даних про вуглецеві викиди з ферм можна провести аналіз та покращити цю сферу людської діяльності з точки зору викидів.

Мета роботи: Публічний доступ до результатів збору інформації про вуглецеві викиди з ферм за допомогою використання технологій блокчейн.

Розділ 1. Проблематика вуглецевих викидів

1.1. Глобальне потепління

Факт існування глобального потепління та співучасть людини в цьому процесі неодноразово ставилися під сумнів не лише «на кухнях», а й на публічних аренах: з вуст популярних та авторитетних вчених, політичних діячів та лідерів думок. Заперечення проблеми та закриття очей на неї завжди найлегше її вирішення, але у випадку, якщо випади в сторону явища справедливі, то всі зусилля щодо мінімізації впливу людини на клімат, в тому числі завдяки мінімізації вуглецевих викидів внаслідок діяльності ферм, а отже і ця робота, втрачають сенс. Тому я хоті би поверхнево розглянути аргументи які підтверджують справедливість переживань людства щодо змін клімату, в тому числі математичні обґрунтування та моделі що підтверджують відповідальність людини у змінах навколишнього середовища.

Основні ознаки глобального потепління:

- Середня температура на землі зростає: починаючи з кінця 19 століття середня температура на землі зросла на 1 градус цельсія, варто зазначити що основний ріст відбувся протягом останніх 40 років.

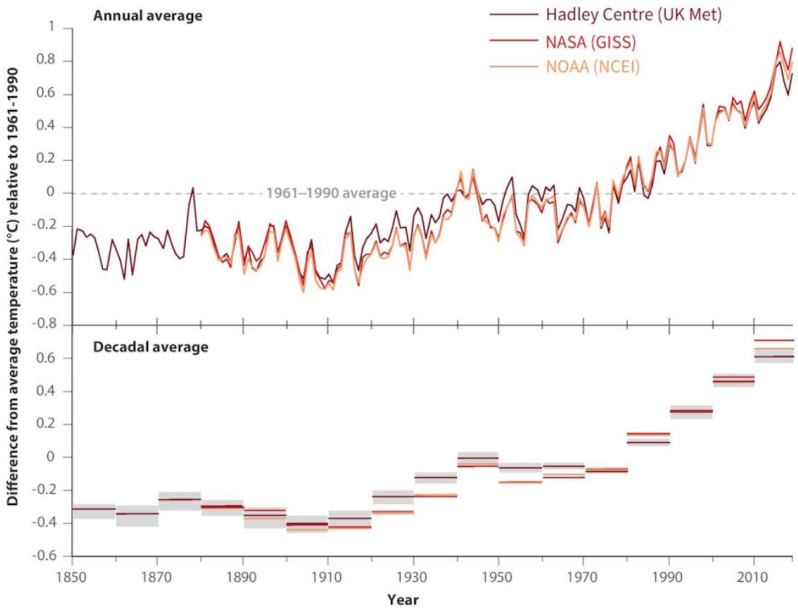

Рис 1.1. Графіку росту середньої температури [8].

- Зменшення об’єму льодових щитів: **NASA** стверджує що у період між 1993 та 2019 роком льодові щити Антарктики та Гренландії зменшувалися щороку на 148 та 279 мільярдів тон щорічно відповідно.

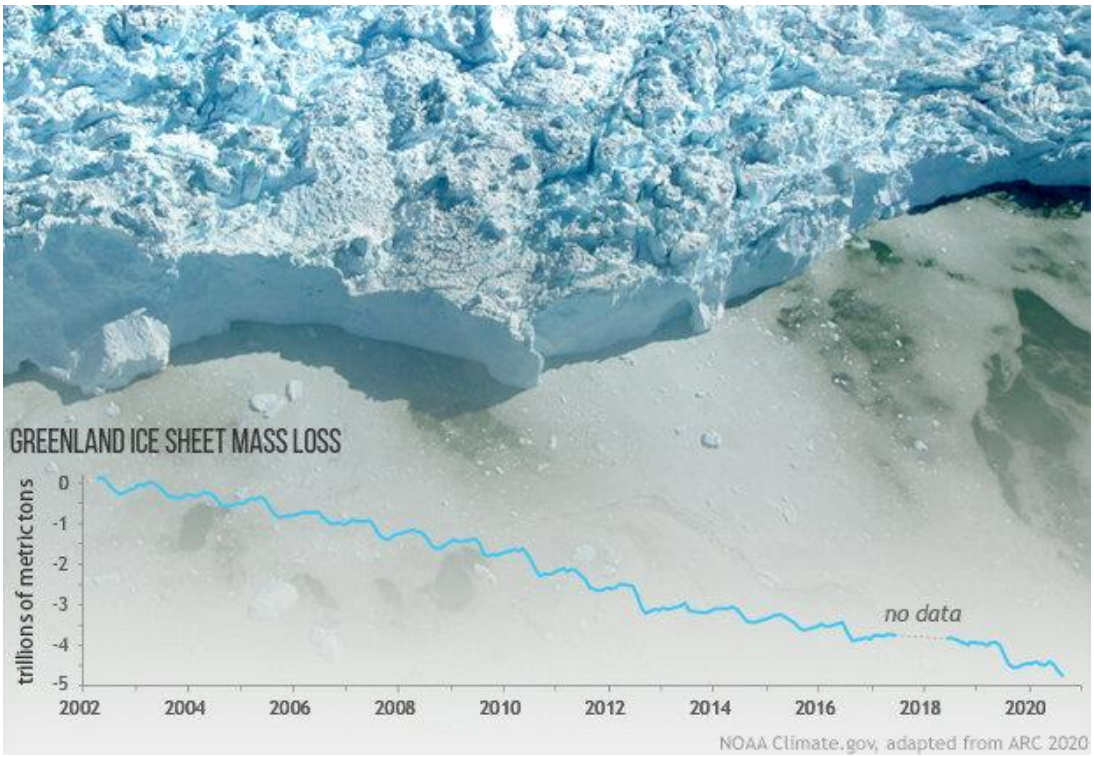

Рис 1.2. Графік падіння об’єму льодових щитів Гренландії [7].

- Ріст рівня моря: За останнє століття рівень моря зріс на 20 см, при чому, як у і випадку з температурою, основний ріст припав на останні десятиліття .

Ці та інші ознаки підтверджують лише факт існування такого явища, оскільки ми можемо об’єктивно оцінити зміни у даних. Проте які причини вважати що людина важлива причина цих змін?

Основні ознаки впливу людини на кліматичні зміни:

- Викиди парникових газів: Людська діяльність спричиняє викиди парникових газів, таких як вуглекислий газ (CO2), метан (CH4), діоксид азоту (N2O) та інші. Ці гази утримують тепло в атмосфері, призводячи до збільшення температури Землі завдяки парниковому ефекту.

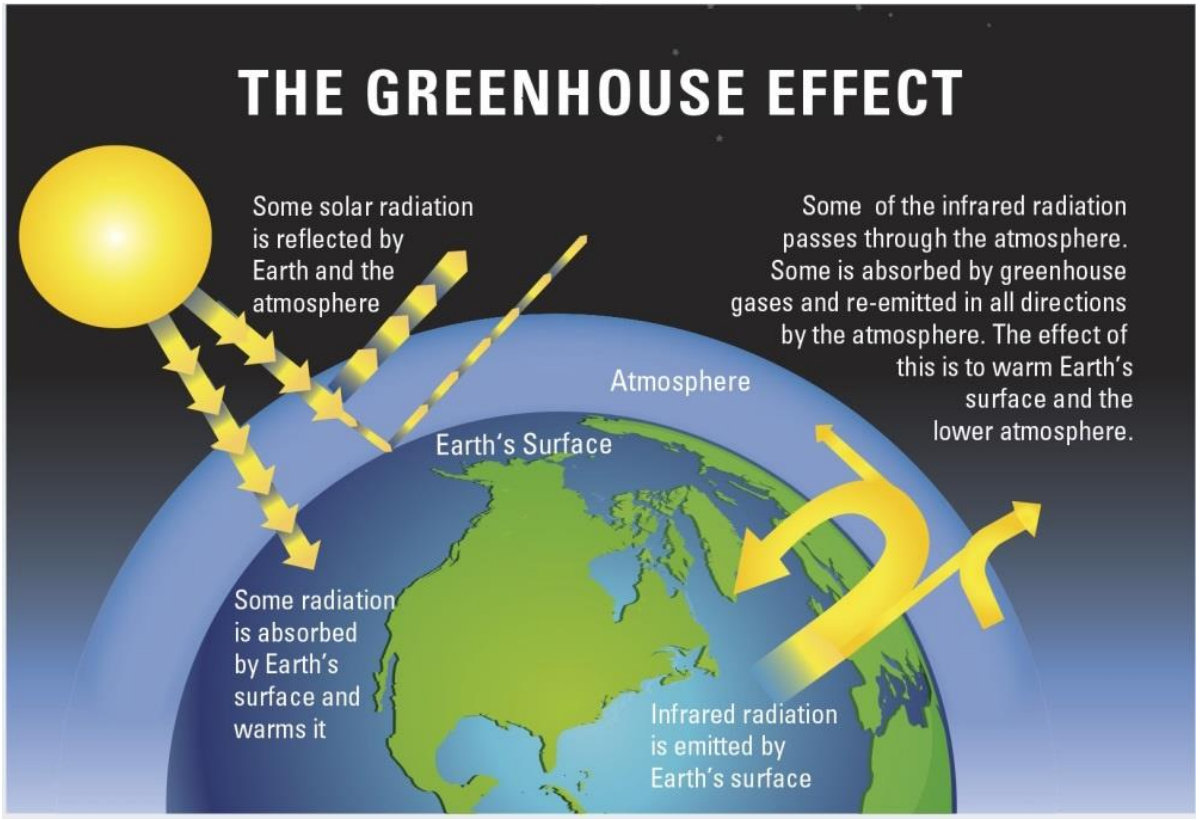

Рис 1.3. Схематичне зображення Парникового ефекту [8].

- Зростання рівня CO2 в атмосфері: Вуглекислий газ є основним парниковим газом, викликаючи близько 75% парникового ефекту. Викиди CO2 з промислових процесів, транспорту, спалювання палива та інших джерел призводять до збільшення його концентрації в атмосфері.

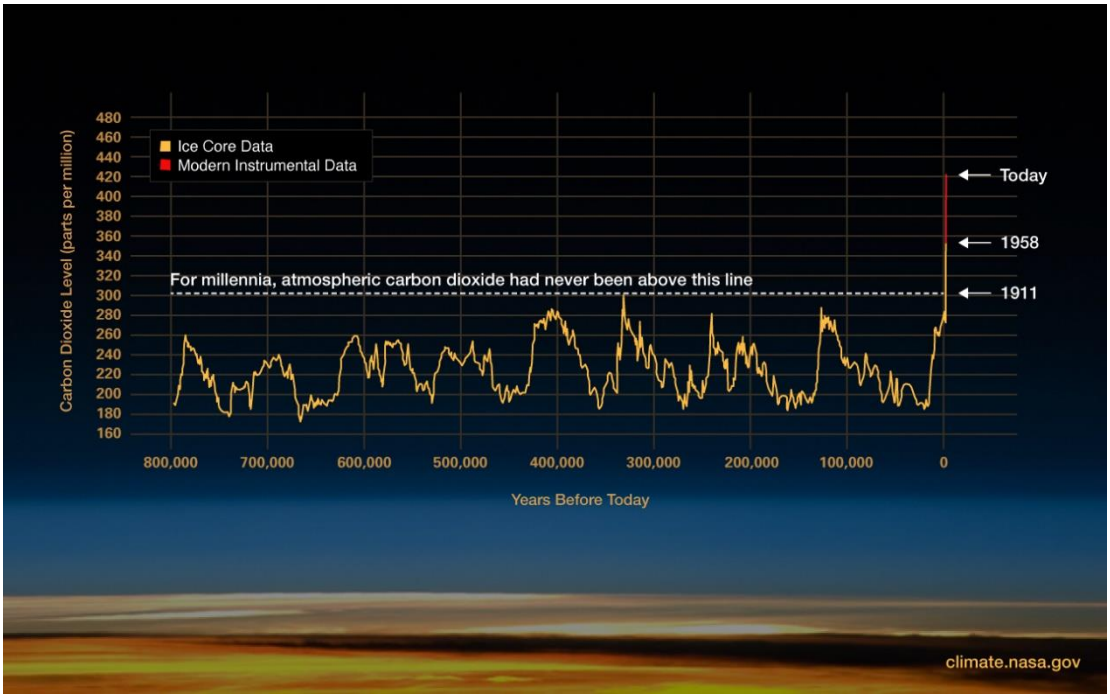

*Рис 1.4. Графік росту **CO2** у повітрі [7].*

- Руйнування лісів і зменшення лісових покривів: Ліси абсорбують CO2 з атмосфери під час фотосинтезу. Знищення лісів через вирубку або пожежі призводить до зменшення здатності землі вбирати вуглекислий газ.

- Забруднення: Забруднення повітря, води та ґрунту також може впливати на клімат. Наприклад, аерозолі в атмосфері можуть впливати на тепловий баланс, а забруднення водних систем може впливати на механізми теплообміну між океанами та атмосферою.

Це основні ознаки, що доводять у великій мірі рукотворність проблеми глобального потепління.

1.2. Вплив вуглецевих викидів на глобальне потепління

Вуглецеві викиди - це загальний термін, який включає різні форми вуглецю, що викидаються в атмосферу внаслідок діяльності людини. Коли говорять про вуглецеві викиди, зазвичай мають на увазі саме викиди CO_2, оскільки вуглекислий газ є найпоширенішим і найважливішим компонентом вуглецевих викидів з точки зору впливу на глобальне потепління, але цей термін також охоплює інші парникові гази, що містять вуглець.

Основні види вуглецевих викидів:

- **Вуглекислий газ (CO_2):** найбільш поширений парниковий газ, який утворюється при спалюванні викопного палива (вугілля, нафти, природного газу, тощо), деревини та інших органічних матеріалів. Він також виділяється під час дихання живих організмів.

- **Метан (CH_4):** Метан є більш потужним парниковим газом, але його концентрація в атмосфері є меншою, ніж у CO_2. Він утворюється під час

видобутку та транспортування вугілля, нафти і природного газу, а також внаслідок ферментації в травному тракті жуйних тварин і на звалищах.

- **Окис азоту (N_2O):** Цей газ також є парниковим і утворюється в основному в сільському господарстві, зокрема через використання азотних добрив.

- **Фторовмісні гази:** Це штучні гази, які мають потужний парниковий ефект і використовуються в холодильниках, кондиціонерах та іншому промисловому обладнанні.

Вуглекислий газ (**CO_2**) є одним з основних парникових газів, що впливають на глобальне потепління. Він затримує теплове випромінювання, що виходить від поверхні Землі, запобігаючи його втечі в космос. Це створює парниковий ефект, який і підвищує температуру планети. Хоча **CO_2** не є єдиним парниковим газом, його концентрація в атмосфері значно зросла через діяльність людини, головним чином через спалювання викопного палива та вирубку лісів.

На даний момент, концентрація **CO_2** в атмосфері становить приблизно 0.041% (410 частин на мільйон). Хоча ця частка і здається незначною, вона має надсуттєвий вплив на кліматичну систему Землі. **CO_2** є одним з головних парникових газів, відповідальних за парниковий ефект. За оцінками, близько 25-30% парникового ефекту створюється саме за рахунок **CO_2**. Водяна пара найбільше впливає на парниковий ефект, але вона є частиною природного циклу, тоді як концентрація **CO_2** значно зросла через антропогенні фактори.

1.3. Парниковий ефект

Вплив людини на клімат в першу чергу спричиняється через існування парникового ефекту. Задля підтвердження загальних висновків з об'єктивних даних наведу дещо спрощене математичне моделювання парникового ефекту

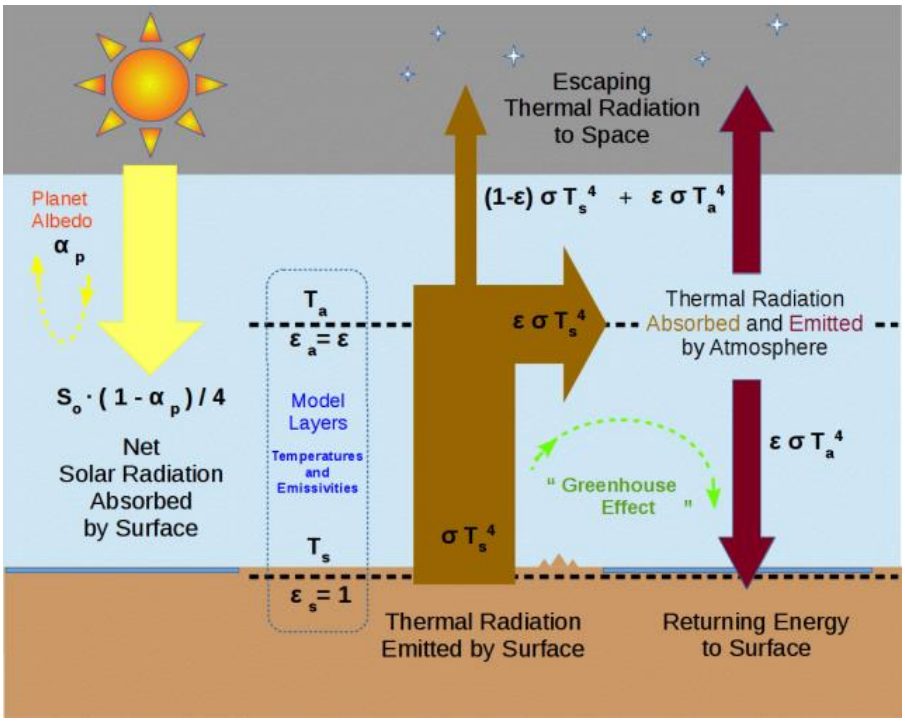

Рис. 1.5. Схематичне зображення роботи парникового ефекту.

Ми шукаємо значення температур Ts і Ta, які зроблять вихідну потужність випромінювання з верхньої частини атмосфери рівною поглиненій потужності сонячного випромінювання. Це означає, що тепло, що поглинається, дорівнює теплу, що випромінюється назад у космос. На Землі вихідне випромінювання є довгохвильовим, а сонячне світло - короткохвильовим. Задля спрощення припустимо, що атмосфера повністю прозора для сонячного світла.

αP=0,30 - Планетарне альбедо (відбиття сонячного світла назад у космос) для Землі.

S0 = 1366 Вт/м² - Сонячна постійна.

Площа поверхні сфери в чотири рази більша за її перетин, тому середнє вхідне випромінювання становить S0/4.

Для довгохвильового випромінювання поверхні Землі використовується закон Стефана-Больцмана [3]:

$$F = \sigma T^4$$

$$\text{Де } \sigma - \text{стала Стефана– Больцмана}$$

який описує, що щільність потоку випромінювання залежить від четвертого ступеня температури поверхні, з урахуванням сталої Стефана-Больцмана. Закон Кірхгофа вказує на те, що атмосфера поглинає та випромінює тепло однаково для будь-якої довжини хвилі в інфрачервоному спектрі. Таким чином, символ є позначає і випромінювальну і поглинальну здатність атмосфери для довгохвильового випромінювання.

Формула для обчислення щільності інфрачервоного потоку з верхньої частини атмосфери [3]:

$$F \uparrow = \epsilon\sigma T_a^4 + (1 - \epsilon)\sigma T_s^4$$

Формула для нульової чистої радіації що залишатиме поверхню Землі [3]:

$$-\frac{1}{4}S_0\left(1 - \alpha_p\right) + \epsilon\sigma T_a^4 + (1 - \epsilon)\sigma T_s^4 = 0$$

Формула для нульового чистого випромінювання, що потрапляє на поверхню [3]:

$$\frac{1}{4}S_0\left(1 - \alpha_p\right) + \epsilon\sigma T_a^4 + \sigma T_s^4 = 0$$

Енергетична рівновага Землі може буде виведена з наведених вище рівнянь [3]:

$$2\epsilon\sigma T_a^4 - \epsilon\sigma T_s^4 = 0$$

Звідси виведемо:

$$T_a = \frac{T_s}{2^{1/4}} = \frac{T_s}{1.189}$$

$$T_s = \left[\frac{S_0(1-\alpha_p)}{4\sigma}\frac{1}{1-\frac{\varepsilon}{2}}\right]^{1/4}$$

Також можна використовувати ефективну температуру випромінювання Te для вирішення цього завдання. Te - це температура, яка відображає щільність вихідного інфрачервоного потоку F, якщо випромінювач був ідеальним. Це означає, що F може бути виражений через Te за допомогою виразу F=σTe^4, де σ - стала Стефана-Больцмана. Цей підхід можна легко уявити у контексті моделі. Також Te може бути рішенням для Ts у випадку, коли ε=0 або атмосфера відсутня [3]:

$$T_e = \left[\frac{S_0(1-\alpha_p)}{4\sigma}\right]^{1/4}$$

Звідси:

$$T_s = \left[T_e\frac{1}{1-\frac{\varepsilon}{2}}\right]^{1/4}$$

Спрощуємо, оскільки припускаємо що $\varepsilon = 1$

T_s=$T_e 2^{1\backslash 4} = 1.189T_e$

$T_{\mathrm{a}} = T_e$

Тепер застосуємо це для Землі

$T_e = 255K = -18C$

Для ε=1

$T_e = 303K = 30C$

При ε=0.78

$T_e = 288.3K$

$T_a = 242.5K$

Що співпадає з твердженням того, що 22% поверхневої енергії потрапляє в космос, та співпадає з опублікованою середньою температурою що становить 287,2 K. Таким чином, ріст ε на 0,02 призведе до підняття середньої температури на 1.2K, для такого росту достатньо подвоєння вуглецю в повітрі.

ε	T_s (K)	T_s (C)
0	254.8	-18.3
0.78	288.3	15.2
0.80	289.5	16.4
0.82	290.7	17.6
1	303.0	29.9

Рис 1.6. Таблиця залежності середньої температури від ε.

Таким чином ми довели, що збільшення кількості вуглецевих викидів напряму призводить до росту середньої температури на планеті вже навіть внаслідок роботи парникового ефекту, а значить потрібно слідкувати за впливом людини на цей параметр, в тому числі внаслідок діяльності ферм.

1.4. Вуглецеві кредити

Одним з методів регуляції викидів є вуглецеві кредити.

Вуглецеві кредити — це інструмент фінансування, що спрямований на зменшення викидів парникових газів та боротьбу зі зміною клімату. Цей механізм базується на ідеї зменшення викидів парникових газів шляхом фінансування проєктів, які сприяють скороченню викидів або збереженню вуглецю в атмосфері.

Принцип дії вуглецевих кредитів зазвичай полягає в наступному: підприємства, організації або проекти, що здійснюють ефективні заходи з мінімізації викидів парникових газів або сприяють вуглецевому зберіганню, можуть отримати кредити або фінансування за рахунок зменшення своїх викидів. Це може бути, наприклад, інвестиція в енергоефективність, відновлювальні джерела енергії, лісові проекти з вуглецевим збереженням тощо.

Отримуючи вуглецеві кредити, організації можуть скоригувати свої викиди та долучитися до глобальних зусиль зі зменшення впливу на клімат

Зазвичай кожен кредит відповідає одній тонні скорочення викидів **CO2** в атмосферу. Такі кредити є актуальним і дієвим методом боротьби з викидами практично у всіх сферах, починаючи з сільського хазяйства і закінчуючи космічними галузями.

Проте є деякі проблеми, що сповільнюють розвиток вуглецевих кредитів:

- **Відсутність взаємодії систем:** це ситуація, коли різні системи не можуть ефективно обмінюватися інформацією та взаємодіяти одна з одною, що ускладнює консолідацію даних в глобальному масштабі.

- **Відсутність прозорості від компаній щодо методів розрахунку:** це ситуація, коли компанії не надають достатньої інформації про те, як саме вони розраховують показники, що стосуються викидів вуглецю або інших аспектів екологічної діяльності.

- **Серйозний ризик подвійного обліку вуглецевого кредиту або "зеленого відмивання":** це ситуація, коли компанії можуть намагатися штучно завищувати свої "зелені" досягнення або отримувати вигоди від кредитів за викиди вуглецю, що не відповідають реальному стану справ.

- **Вимога прозорості фінансових та інституційних інфраструктур на вуглецевому ринку:** це необхідність налагодження механізмів, що забезпечують доступність інформації та довіру до фінансових операцій і структур, пов'язаних із зменшенням викидів вуглецю та іншими екологічними заходами.

- **Регуляторні проблеми:** це проблеми, пов'язані з неефективністю або недостатньою дієвістю законодавства та регуляційних механізмів, що регулюють вуглецевий ринок, що може призвести до маніпуляцій, шахрайства або недостатньої контролю за спекуляціями.

Одним з шляхів вирішення цих проблем та мінімізації ризиків є протокол **Axone**, що дозволяє автоматизувати збір, обробку та аналіз даних, спрощуючи складні робочі процеси. Завдяки **Axone** витрати енергії, транспорт, промислові процеси - дані з різних джерел, можна легко збирати та агрегувати.

Завдяки прозорості протоколу стає можливим відслідковувати кожен етап та бути впевненим у правдивості даних, що дозволяє підняти рівень довіри обидвох сторін. Такий підхід дозволяє ефективно знаходити місця можливого зменшення рівню викидів, приймати практичні бізнес рішення та обмінюватися даними.

Робота з **Axone** складається з таких основних етапів:

- **Збір даних з різних джерел:** ЕОМ, корпоративна соціальна відповідальність, серед учасників таких як постачальники чи клієнти та форматів серед яких датчики Інтернету речей, лічильники, системи енергоменеджменту.

- **Агрегація та нормалізація:** процес об'єднання та стандартизації даних з різних джерел для забезпечення їхньої якості та послідовності перед подальшою обробкою. Для цього проводиться агрегація, нормалізація, конвертація, стандартизація та перевірка якості даних.

- **Обробка та розрахунок:** етап, на якому здійснюється аналіз та розрахунок викидів вуглецю за різними категоріями, що визначені відповідними протоколами.

- **Аналіз та звітність:** проводиться додатковий аналіз на основі використання результатів, детальна та автоматизована звітність для ідентифікації значущих джерел, тенденцій та пошуку можливостей для зменшення викидів. Надання відкритого доступу для громадських та економічних учасників.

- **Компенсація:** обмін вуглецевими даними, верифікація та реєстрація у офіційному реєстрі, що дозволяю уникнути подвійного обліку кредитів паралельно. **Axone** забезпечує постійний моніторинг за межами обчислень в певний момент часу для виявлення тенденцій та впровадження заходів.

Протокол **Axone** має перспективи стати широко використовуваним, стандартизованим та передовим інструментом для розрахунку та управління вуглецевими кредитами. У перспективі його вплив може вийти за межі окремих організацій, допомагаючи світовим зусиллям у боротьбі зі зміною клімату та досягненні сталого розвитку. Водночас протокол спонукає організації ділитися якіснішими даними та розвивати нові бізнес-моделі.

Розділ 2. Огляд технологій Smart Data Space і OKP4

2.1. Smart Data Space

Smart Data Space – місце для зберігання і структурування **Smart Data Models**.

Smart Data Models - це спільна ініціатива, яка розвивається за підтримки Фонду **FIWARE**, **TMForum**, **IUDX** та **OASC**, а також інших осіб та організацій. Ця програма створює безкоштовні відкриті моделі даних з відкритими ліцензіями, що дозволяють безкоштовне використання, модифікацію та обмін модифікаціями.

Місія **Smart Data Models** - це сприяння спільному обміну даними шляхом створення стандартних та універсальних моделей даних. Ці моделі можуть використовуватися різними організаціями та системами для забезпечення легкого та ефективного обміну інформацією та співпраці.

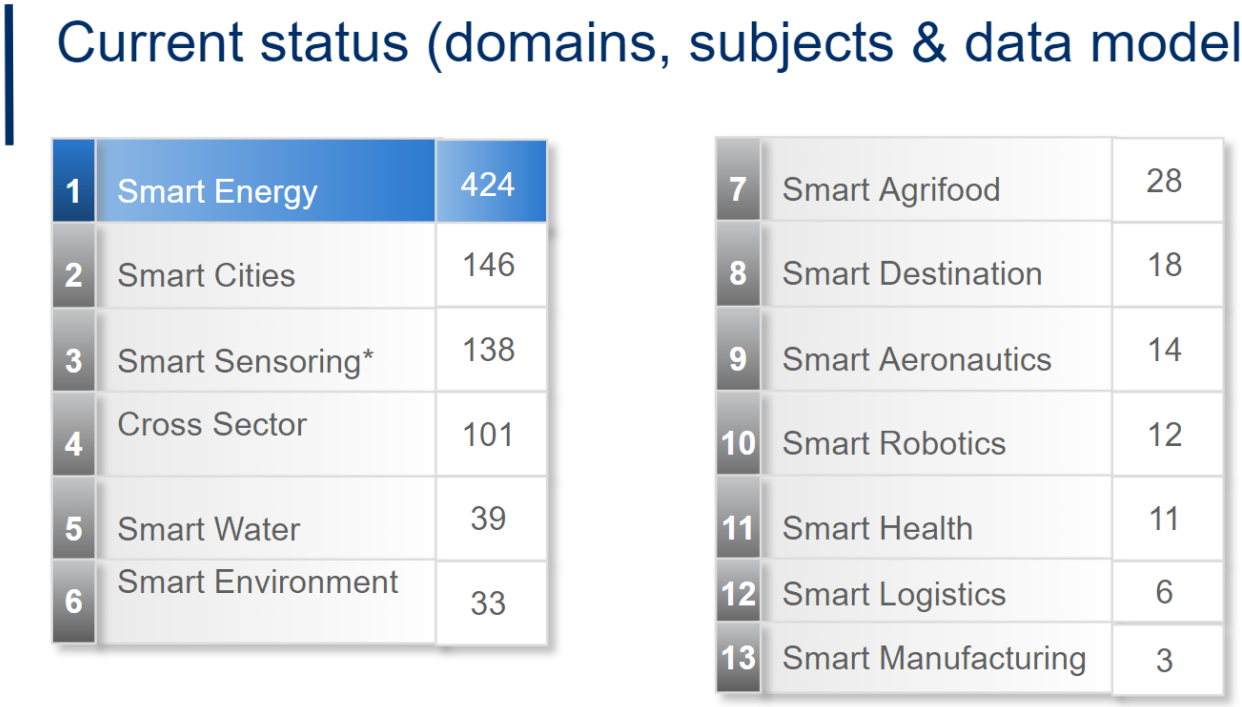

Рис 2.1.1. Актуальна кількість доступних моделей за кожною з категорій [11].

2.2. Основи блокчейну

2.2.1. Блокчейн

Блокчейн - це технологія розподіленого реєстру, яка дозволяє зберігати інформацію в захищеній, незмінній та прозорій формі. Вона складається з послідовних блоків, які містять набір даних про транзакції. Кожен блок зв'язаний з попереднім за допомогою криптографічних хешів, створюючи ланцюжок (блокчейн), що унеможливлює зміну даних без виявлення.

Основні характеристики блокчейну:

- **Розподілений реєстр**: Дані зберігаються одночасно на багатьох комп'ютерах (вузлах) у мережі. Кожен вузол має копію всього ланцюжка блокчейну.

- **Децентралізація**: Блокчейн не має центрального органу керування. Всі рішення приймаються консенсусом між вузлами. Такий підхід забезпечує високу надійність і безпеку, оскільки зміна даних на одному вузлів не впливає на інші.

- **Незмінність**: Після додавання блоку до блокчейну його неможливо змінити або видалити. Незмінність блоків забезпечується криптографічними хешами, які пов'язують кожен блок з попереднім.

Кожен блок у блокчейні містить дві основні частини: заголовок блоку і тіло блоку:

Заголовок блоку (Block Header) містить:

- **Хеш попереднього блоку:** Це хеш попереднього блоку в ланцюзі, ця властивість забезпечує зв'язок між блоками.

- **Власний хеш блоку:** Це унікальний хеш, який представляє дані поточного блоку і створюється за допомогою криптографічних алгоритмів. Блокчейни часто послуговуються алгоритмом хешування **SHA-256.**

- **Часова мітка:** Час, коли блок був створений.

- **Номер версії:** Версія програмного забезпечення, яка використовується для створення блоку.

- **Нонсенс (nonce):** Число, яке використовується для майнінгу блоку.

Тіло блоку (Block Body) містить:

- **Дані про транзакції:** Список транзакцій, які були підтверджені і включені до блоку.

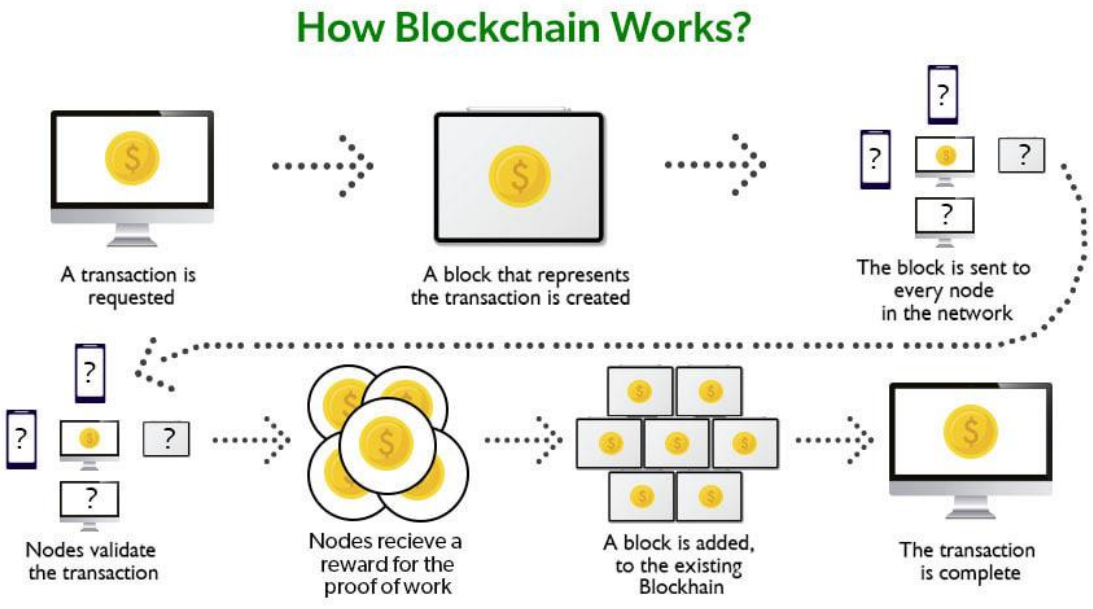

Рис 2.2.1. Схематичне зображення роботи блокчейну [17].

Є два основні механізми створення нових блоків – майнінг та алгоритми консенсусу:

Майнінг (для PoW блокчейнів):

- **Процес майнінгу:** Майнер (учасник мережі) збирає непідтверджені транзакції і створює новий блок. Щоб додати блок до блокчейну, майнер повинен вирішити складну криптографічну задачу - знайти такий нонсенс, який разом з іншими даними блоку буде давати хеш з певною кількістю початкових нулів.

- **Нагорода за майнінг:** Перший майнер, який вирішить задачу, отримує винагороду у вигляді новостворених монет та комісій за транзакції у блоці. Після цього новий блок додається до блокчейну, і копія оновленого блокчейну розповсюджується по всій мережі.

Алгоритми консенсусу:

- **Proof of Work (PoW):** Вимагає від майнерів виконання обчислювальної роботи для додавання нового блоку. Цей процес дуже енергоємний, але забезпечує високу безпеку.

- **Proof of Stake (PoS):** Вимагає від учасників володіння певною кількістю криптовалюти. Власники "ставлять" свою криптовалюту, щоб мати шанс створити новий блок і отримати нагороду. Це менш енергоємний метод, ніж PoW. Саме на такому алогоритмі побудований **Axone.**

2.2.2. Хешування SHA-256

Хешування — це процес перетворення даних будь-якого розміру на фіксованого розміру хеш за допомогою спеціальних криптографічних алгоритмів. Один з найбільших відомих алгоритмів хешування є **SHA-256** (**Secure Hash Algorithm 256-bit**), який часто використовується в блокчейнах, таких як **Bitcoin**.

Кожен блок у блокчейні має свій унікальний хеш, який генерується на основі вмісту блоку. Цей хеш виконує кілька важливих функцій:

- **Ідентифікація блоку**: Унікальний хеш блоку дозволяє легко ідентифікувати конкретний блок у блокчейні.

- **Інтеграція даних**: Хешування забезпечує цілісність даних. Якщо хоча б один біт даних у блоці змінюється, хеш змінюється повністю, що робить неможливою будь-яку підміну даних.

- **Зв’язок з попереднім блоком**: Кожен блок містить хеш попереднього блоку, це створює ланцюжок, за допомогою якого всі блоки пов’язуються.

Переваги хешування в блокчейні:

- **Безпека**: Хешування захищає дані від несанкціонованих змін.

- **Цілісність**: Будь-яка зміна даних в блоці змінить хеш, що дозволяє легко перевірити і впевнитися в справжності даних.

- **Прозорість і довіра**: Оскільки всі блоки пов’язані, кожен учасник мережі може перевірити цілісність і порядок блоків.

SHA-256 (Secure Hash Algorithm 256-bit) — це криптографічний алгоритм хешування, який перетворює вхідні дані довільного розміру на фіксований 256-бітовий (32-байтовий) хеш. [20]

Основні характеристики алгоритму:

- **Фіксований розмір хешу**: Незалежно від розміру вхідного повідомлення, вихід завжди має розмір 256 біт.

- **Односторонність**: Неможливо відновити початкове повідомлення з його хешу.

- **Широкий розподіл**: Маленька зміна вхідного повідомлення призводить до значних змін хешу.

Розглянемо роботу алгоритму:

1. Попередня обробка:

Додавання заповнювачів: Вхідні дані доповнюються одиничним бітом 1, після чого додається достатня кількість нулів, щоб довжина результату стала кратною - 512 бітам, за винятком останніх 64 бітів.

Заповнення довжини повідомлення: Останні 64 біти заповнюються двійковим представленням початкової довжини вхідного повідомлення (у бітах).

2. Декомпозиція блоку:

Для кожного блоку M ∈ {0, 1}^512, 64 слова по 32 біти кожне конструюються наступним чином:

перші 16 отримуються шляхом розділення M на блоки по 32 біти [20]:

$$M = W1||W2|| \cdots ||W15||W16$$

решта 48 отримуються за формулою:

$$Wt = \sigma_1(Wt-2) + Wt-7 + \sigma_0(Wt-15) + Wt-16,\ 17 \leq t \leq 64.$$

σ_0 і σ_1 — це побітові операції:

$$\sigma_0(x) = (x \gg 7) \oplus (x \gg 18) \oplus (x \gg 3)$$
$$\sigma_1(x) = (x \gg 17) \oplus (x \gg 19) \oplus (x \gg 10)$$

де:

$\gg$ - циклічний зсув вправо

$\oplus$ - побітове виключне АБО (XOR).

3. Обчислення Хешу:

Використовуються вісім початкових констант (фіксовані хеш-значення), які представляють перші 32 біти дробових частин квадратних коренів перших восьми простих чисел (2, 3, 5, 7, 11, 13, 17, 19) [20]:

$$a_2\text{^}(0) = 0x6a09e667$$
$$b_3\text{^}(0) = 0xbb67ae85$$
$$c_5\text{^}(0) = 0x3c6ef372$$
$$d_7\text{^}(0) = 0xa54ff53a$$
$$e_{11}\text{^}(0) = 0x510e527f$$

$$f_{13}\hat{}(0) = 0x9b05688c$$

$$g_{17}\hat{}(0) = 0x1f83d9ab$$

$$h_{19}\hat{}(0) = 0x5be0cd19$$

Наступним кроком вісім 32-бітних змінних (A, B, C, D, E, F, G, H) ініціалізуються значеннями з початкового хешу [20]:

$$(a, b, c, d, e, f, g, h) = (a_2\hat{}(t-1), b_3\hat{}(t-1), c_5\hat{}(t-1), d_7\hat{}(t-1), e_{11}\hat{}(t-1), f_{13}\hat{}(t-1), g_{17}\hat{}(t-1), h_{19}\hat{}(t-1))$$

Основний цикл обробки 64 кроків для кожного блоку використовує функції змішування та додавання [20]:

$$T1 = h + \Sigma_1(e) + Ch(e, f, g) + Ki + Wi$$

$$T2 = \Sigma_0(a) + M\,aj(a, b, c)$$

$$h = g$$

$$g = f$$

$$f = e$$

$$e = d + T1$$

$$d = c$$

$$c = b$$

$$b = a$$

$$a = T1 + T2$$

де:

$\Sigma_0(x) = (x \gg 2) \oplus (x \gg 13) \oplus (x \gg 22)$

$\Sigma_1(x) = (x \gg 6) \oplus (x \gg 11) \oplus (x \gg 25)$

Ch(x, y, z) = (x & y) ⊕ (¬x & z)

Maj(x, y, z) = (x & y) ⊕ (x & z) ⊕ (y & z)

Kt - константи, що використовуються для кожного кроку і представляють перші 32 біти дробових частин кубічних коренів перших 64 простих чисел.

Після чого обчислюються нові значення початкових констант [20]:

$$a_2^{\wedge}(t) = a_2^{\wedge}(t-1) + a$$
$$b_3^{\wedge}(t) = b_3^{\wedge}(t-1) + b$$
$$c_5^{\wedge}(t) = c_5^{\wedge}(t-1) + c$$
$$d_7^{\wedge}(t) = d_7^{\wedge}(t-1) + d$$
$$e_{11}^{\wedge}(t) = e_{11}^{\wedge}(t-1) + e$$
$$f_{13}^{\wedge}(t) = f_{13}^{\wedge}(t-1) + f$$
$$g_{17}^{\wedge}(t) = g_{17}^{\wedge}(t-1) + g$$
$$h_{19}^{\wedge}(t) = h_{19}^{\wedge}(t-1) + h$$

Після обробки всіх блоків остаточні значення з’єднуються для формування хешу. Хеш значення є конкатенацією змінних a_2N, b_3N, c_5N, d_7N, $e_{11}N$, $f_{13}N$, $g_{17}N$, $h_{19}N$ після обробки останнього блоку [20]:

$$H = a_2^{(N)} \| b_3^{(N)} \| c_5^{(N)} \| d_7^{(N)} \| e_{11}^{(N)} \| f_{13}^{(N)} \| g_{17}^{(N)} \| h_{19}^{(N)}$$

У результаті бачимо, що алгоритм SHA-256 використовує складні побітові операції та криптографічні функції для створення надійного та стійкого до колізій хешу. Це робить його ідеальним для використання в блокчейнах та інших системах, де потрібна висока безпека даних.

2.2.3. Смарт контракти

Смарт-контракти - це програми, що зберігаються на блокчейні, які автоматично виконують умови договору при виконанні певних умов. Вони дозволяють автоматизувати процеси проводячи транзакції і угоди без третіх осіб, та забезпечують високу надійність і безпеку угод [18].

Основні переваги смарт-контрактів:

- **Автоматизація:** Смарт-контракти виконуються автоматично при дотриманні визначених умов.

- **Безпека:** Завдяки використанню блокчейн-технології смарт-контракти захищені від підробок і втручання.

- **Прозорість:** Умови та результати виконання смарт-контрактів доступні всім учасникам мережі.

- **Незмінність:** Після того, як смарт контракт записаний в блокчейн, його неможливо змінити. Це забезпечує високий рівень безпеки та довіри до контракту.

- **Економічна ефективність:** Використання смарт контрактів знижує витрати на ведення бізнесу, оскільки вони усувають необхідність у посередниках і зменшують витрати на аудит.

Коли дві сторони укладають угоду, вони записують умови в смарт контракт. Ці умови можуть бути як простими, наприклад, "якщо товар доставлено, то перевести кошти продавцю", так і складними, складаючись з кількох взаємопов’язаних дій і перевірок.

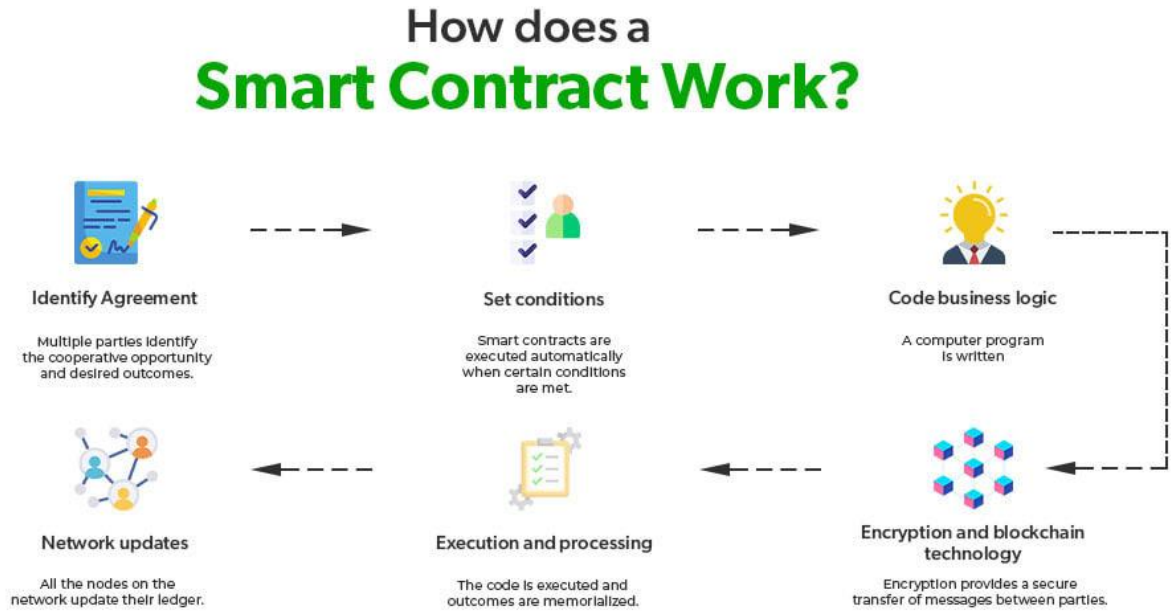

Рис 2.2.2. Схематичне зображення роботи смарт контрактів [18].

Після написання смарт контракт розгортається в блокчейні. Це означає що він стає частиною незмінної розподіленої бази даних. Усі учасники мережі можуть бачити і перевіряти відповідний контракт, що в свою чергу гарантує прозорість та відкритість. Коли виконуються умови, прописані в контракті, він автоматично виконує свої дії, наприклад, переводить гроші з одного рахунку на інший або видає дозвіл на доступ до даних.

Наприклад: у випадку купівлі-продажу нерухомості, смарт контракт може автоматично перевести право власності на будинок від продавця до покупця після отримання платежу. При цьому всі транзакції та дії записуються в блокчейн, що виключає необхідність у посередниках, таких як нотаріуси або банки, знижує витрати і ризик людських помилок.

Смарт контракти також забезпечують безпеку, оскільки вони працюють в розподіленому середовищі, де кожна зміна фіксується і не може бути змінена заднім числом без згоди учасників. Це особливо важливо для забезпечення виконання угод та захисту від шахрайства. Також смарт контракти можуть бути використані в різних галузях, таких як фінанси, логістика, страхування, де вони автоматизують процеси і підвищують їх ефективність.

2.3. AXONE (OKP4)

Axone (Колишня назва платформи - *OKP4)* - це публічний блокчейн на основі механізму консенсусу доказу частки володіння, побудований для спільного використання даних із мінімальним рівнем довіри [13].

Спільне використання даних із мінімальним рівнем довіри це можливість використовувати дані для створення корисних знань і додатків, не розголошуючи самі дані іншим користувачам. Це досягається за допомогою безпечної мережі, де дані залишаються конфіденційними, і користувачі мають доступ лише до оброблених результатів, а не до початкових даних. Такий підхід дозволяє обмінюватися цінностями без обмежень та ризиків, пов'язаних з витоком даних.

Цей блокчейн відкритий та здатний взаємодіяти з іншими системами. **Axone** надає можливість створювати розумні контракти з високою ступенем налаштування, а також підтримує механізми управління та моделі бізнесу, що дозволяють узгоджувати інтереси між учасниками та розробляти корисні розподілені додатки.

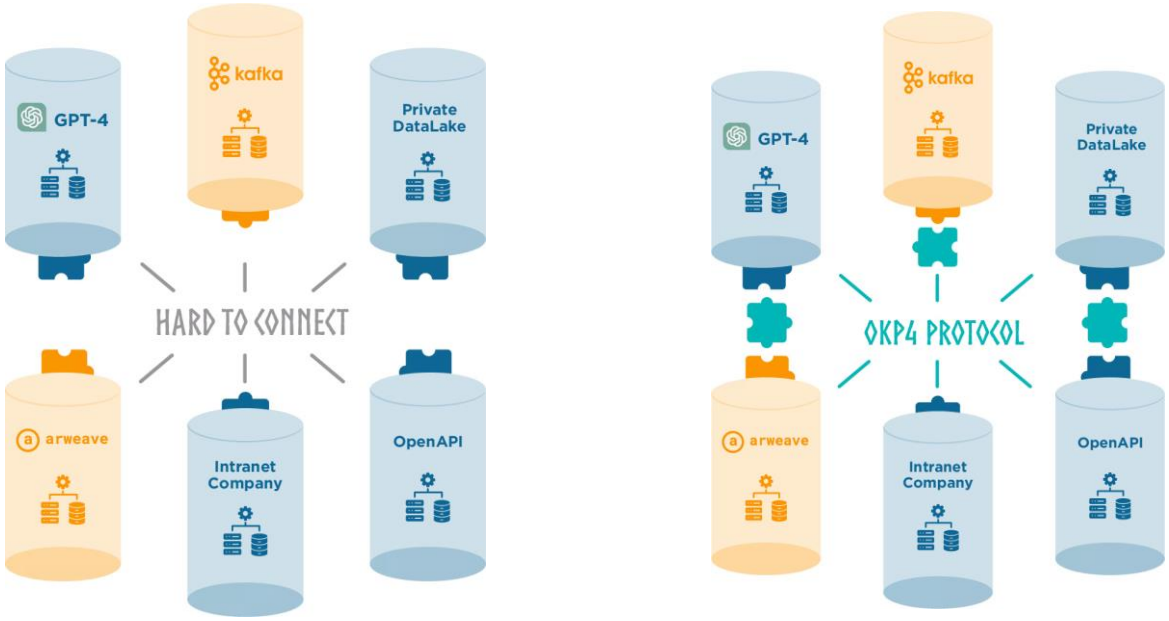

Рис 2.3.1. Схематичні рекламні зображення, що демонструють як Axone дозволяє поєднувати між собою різні системи [13].

Axone гарантує довіру завдяки децентралізації, пропонує гнучку систему стимулювання та зменшує фрагментацію цифрової екосистеми, дозволяючи використовувати "все як сервіс". **Axone** використовує Зони для організації ресурсів та стандартизованого тлумачення елементів за допомогою онтології на ланцюжку. Він також здійснює оркестрацію робочих процесів на спільних ресурсах за допомогою розумних контрактів та модулів для будь-яких додатків.

Axone вирішує проблему обміну цифровими ресурсами, яка полягає в тому, що сучасні дані зберігаються в ізольованих сховищах, через що їх потенціал залишається невикористаним через відсутність довіри та стимулів для обміну. Ці умови спричинені відсутністю інфраструктури, що дозволяла б легко обмінюватися ресурсами та узгоджувати правила взаємодії між ними. **Axone** ж створює блокчейн, який дозволяє ефективно обмінюватися цифровими ресурсами, забезпечуючи аудит та дотримання правил згоди, що відкриває шлях до більш ефективного використання цих ресурсів та збільшення їхньої цінності.

Приклади використання:

- **Сільське господарство:** оптимізація поливу, управління посівами за допомогою супутників, покращення догляду за ґрунтами.

- **Охорона здоров'я:** персоналізована медицина, розвиток боротьби з раком, онлайн медицина, епідеміологія.

- **Штучний інтелект:** створення баз даних та обчислювальних мереж для машинного навчання, створення інструментів для машинного навчання здатного забезпечувати конфіденційність використаних даних, керування моделями.

- **Промисловість:** автоматизування, утилізація викидів, слідкування за якістю продуктів, симуляція процесів, оптимізація виробництва.

2.4. Принцип роботи AXONE

2.4.1. Онтології

Онтологія - це визначення структури даних та взаємозв'язків між ними в контексті криптовалютного середовища. Іншими словами це спрощена модель світу, що описує об'єкти, їх атрибути та зв'язки, допомагає стандартизувати і узгоджувати дані між різними блокчейнами, протоколами та смарт-контрактами [16]

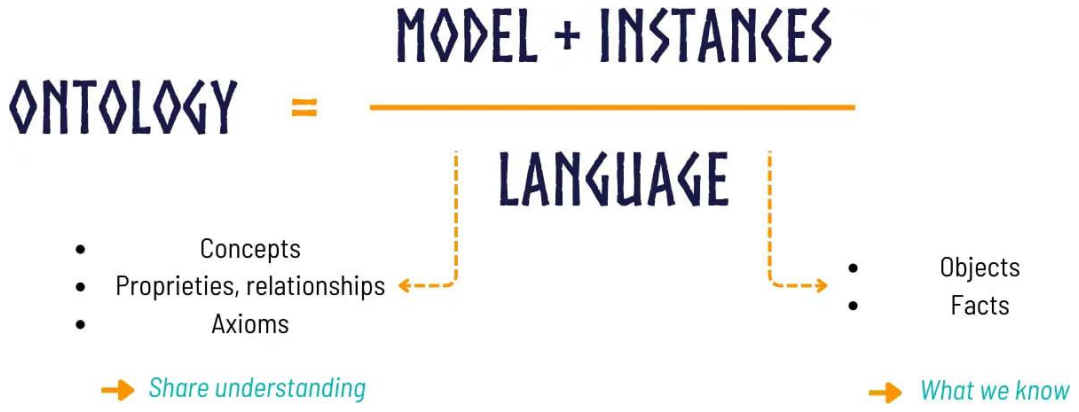

Рис 2.4.1. Схематичне зображення яке пояснює що таке Онтологія [16].

У протоколі **Axone** використання онтологій є надзвичайно важливим, оскільки вони є основним інструментом для докладного опису простору даних, включаючи його найбільш складні аспекти. Онтології забезпечують детальне уявлення як про набори даних, так і сервіси, що значно спрощує загальне розуміння системи. Крім того, вони встановлюють зв'язок між управлінням та оркестрацією всередині мережі даних.

Онтології дозволяють досягти:

- ***Стандартизації термінології*:** завдяки стандартизації вдається запобігти непорозумінь між учасниками

- ***Взаємозамінності систем і інструментів:*** продумана та якісна онтологія дозволяє забезпечити взаємодію між різними системами та інструментами, що полегшує обмін знаннями між різними сторонами та дозволяє взаємозаміняти системи та інструменти

- ***Покращення дослідження та аналізу даних:*** завдяки точному опису взаємозв'язків і концепцій у певній області вдається покращити якість аналізу даних

- **Структурування даних:** дані організовані логічно, послідовно та систематично, завдяки чому спрощується доступ до них, обробка та аналіз.

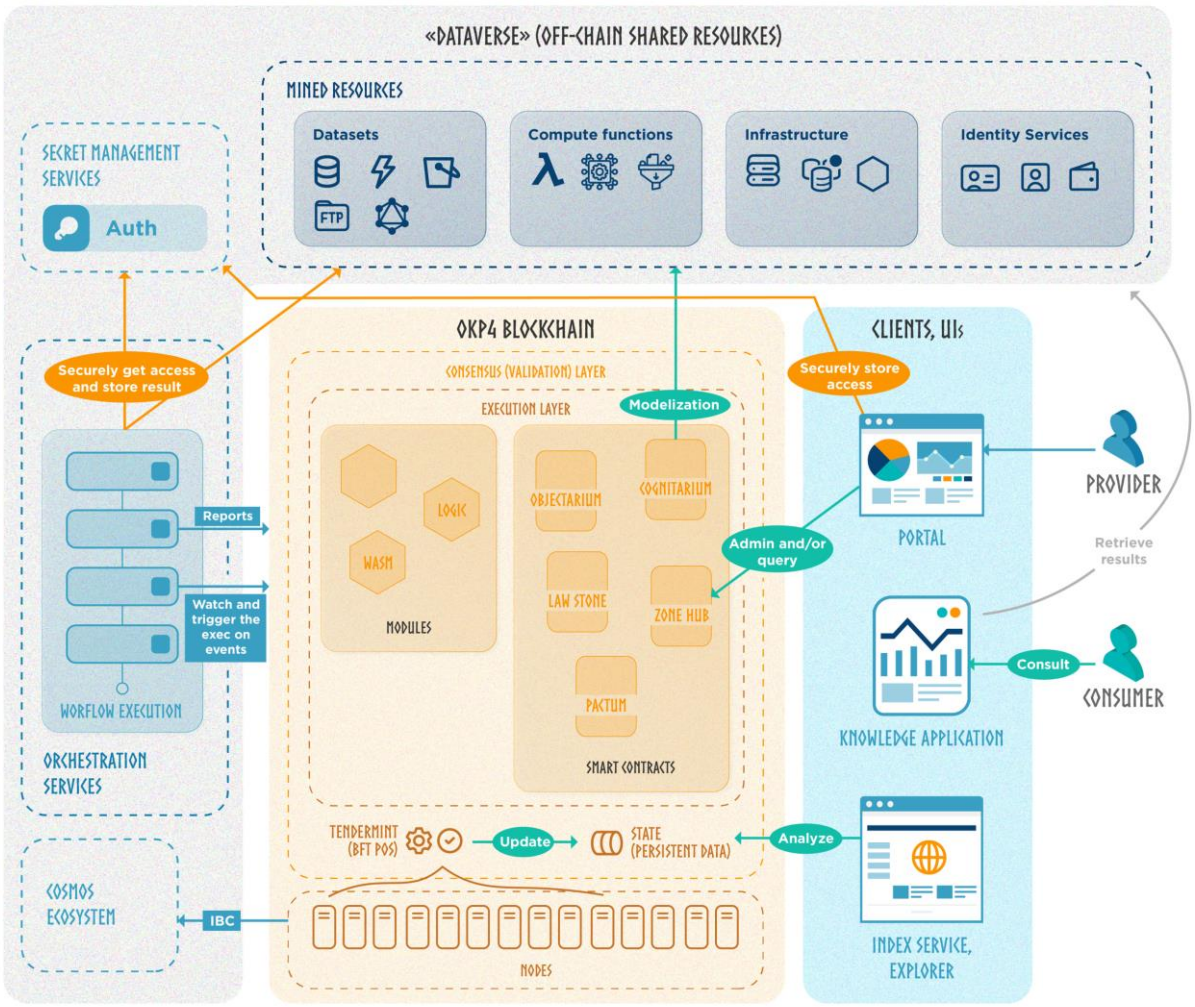

Рис 2.4.2. Схематичне зображення архітектури ***Axone*** *[13].*

2.4.2. Управління

У протоколі **Axone** під управлінням розуміється створення та впровадження правил, політик та процесів, які визначають, як цифрові ресурси мережі даних повинні взаємодіяти та функціонувати. Це схоже на створення набору правил гри, за якими всі гравці повинні діяти. Управління є основою для ефективної організації та контролю різних аспектів даних у мережі, що допомагає забезпечити їхню надійність та ефективність.

Чотири головні джерела управління в Axone:

- **Управління протоколом: Axone** встановлює основні правила та регламенти, що керують його блокчейном. Це надає основу для всіх інших рівнів та визначає основні принципи його функціонування.

- **Правила зони:** Правила зони встановлюють межі певної області у дата-всесвіті, де діють правила управління. Кожна зона має свій набір правил, призначених для її конкретної функції.

- **Згода на ресурси:** Згоди визначають угоди щодо використання ресурсів у межах зон. Це включає цифрові ресурси та послуги, і дозволяє встановлювати межі та умови для їхнього використання.

- **Угоди:** Угоди між сторонами створюють спільне розуміння та узгодження стосовно майбутніх дій або результатів. Це означає, що сторони добровільно приймають та дотримуються умов, викладених угодою, демонструючи свою готовність до співпраці та повагу до прав та обов'язків, визначених у контракті.

Рис 2.4.3. Схематичне зображення різних рівнів управління [13].

Управління протоколом - це головний керівний механізм, який визначає напрями розвитку, ступінь децентралізації та рівень безпеки мережі. Воно стимулює прогрес, дозволяючи вносити зміни, сприяючи інноваціям та змагальності. Також це може сприяти більшій демократії, залучаючи учасників у прийняття рішень. Управління також гарантує надійність мережі, захищаючи її від вразливостей та атак.

Правила зони - це набір правил, які визначають умови взаємодії цифрових ресурсів та суб'єктів у певній області. Вони забезпечують відповідність і раціональне використання ресурсів в межах конкретного контексту. Це дозволяє забезпечити ефективне управління та дотримання правил у визначеній зоні.

Багато правил можуть описувати різні характеристики суб'єкта дії. Ці правила можуть бути визначені в спеціальних даних про об'єкт або підтверджені, перевіривши достовірність інформації.

Розділ 3. Інтеграція Smart Data Space з блокчейн-платформою OKP4

3.1. Завантаження необхідних даних:

3.1.1. Авторизація та отримання токена

У сучасних веб-додатках і **API** безпека є однією з найважливіших складових. Управління ідентифікацією та доступом (**IAM**) є критичним для забезпечення безпеки додатків і даних користувачів. В якості безпекового інструмента був обраний **Keyrock**, компонент системи **FIWARE**, що дозволяє ефективно управляти ідентифікацією та доступом, забезпечуючи аутентифікацію та авторизацію користувачів.

Keyrock — це сервер управління ідентифікацією та доступом, який підтримує протокол **OAuth2**. Він дозволяє керувати користувачами, ролями, дозволами та організаціями. **Keyrock** забезпечує аутентифікацію користувачів і видачу токенів доступу, які використовуються для авторизації запитів до **API**.

Ми використовуємо **Keyrock** задля забезпечення:

- Авторизації доступу до ресурсів на основі ролей і дозволів.
- Аутентифікації користувачів за допомогою логіна та пароля.
- Управління користувачами та їхніми ролями в організаціях.

Розглянемо процес отримання **OAuth2** токену з використанням мови програмування **Python** в середовищі **Google Colab:**

```
import requests
import json
import http.client

def get_oauth2_token():
```

```
    url = "https://keyrock.nosc-ua-dih.app/oauth2/token"
    headers = {
        "Content-Type": "application/x-www-form-urlencoded"
    }
    data = {
        "username": "********@test.com",
        "password": '************',
        "grant_type": "password",
        "client_id": "9486fd82-b235-4307-8016-************",
        "client_secret": "22c78fc0-4874-4698-965c-*************"
    }
    response = requests.post(url, headers=headers, data=data)
    if response.status_code == 200:
        token = json.loads(response.text)["access_token"]
        return token
    else:
        return None

if __name__ == "__main__":
    token = get_oauth2_token()
    if token:
        print("Access token:", token)
    else:
        print("Failed to get access token")
```

Код 3.1.1. Процес отримання OAuth2 токену за допомогою клієнтських облікових даних у Python.

Перш за все імпортуємо необхідні для відправки **HTTP**-запитів та роботи з **JSON**-даними бібліотеки:

```
import requests
import json
import http.client
```

Розглянемо функцію **get_oauth2_token**:

url – встановлюємо **URL** для запиту токену.

headers – Встановлюємо заголовок **Content-Type**, таким чином зазначаємо, що дані передаються у форматі **application/x-www-form-urlencoded**.

data – вказуємо необхідні для отримання токена дані, а саме логін та пароль, тип авторизації, ідентифікатор клієнта та його секрет.

Використовуючи **requests.post** відправляємо запит для отримання токена. В тіло запиту передаємо необхідні дані. Обробляємо результат, у разі успішного виконання коду повертаємо очікуваний токен, або ж **None** у випадку збою.

У разі успішного виконання методу **get_oauth2_token** виводимо на екран зміст токена, у зворотному випадку повідомляємо користувача що токен не вдалося отримати.

3.1.2. Виконання HTTPS запитів з використанням отриманого токену

```
conn = http.client.HTTPSConnection("ds1.nosc-ua-dih.app")

payload = ""

headers = {
    'User-Agent': "insomnia/9.1.1",
    'X-Auth-token': get_oauth2_token(),
    'Link': "<https://raw.githubusercontent.com/smart-data-
models/dataModel.Agrifood/master/context.jsonld>;"
    }

conn.request("GET", "/ngsi-ld/v1/types", payload, headers)

res = conn.getresponse()
data = res.read()

print(data.decode("utf-8"))
```

Код 3.1.2. Процес використання отриманого токена для виконання запиту.

Створюємо **HTTPS**-з'єднання з сервером **ds1.nosc-ua-dih.app** за допомогою **http.client.HTTPSConnection**:

```
conn = http.client.HTTPSConnection("ds1.nosc-ua-dih.app")
```

Формуємо заголовки запитів, вказуємо **User-Agent**, отриманий раніше токен та посилання на дані. Відправляємо **GET**-запит до ендпоінту **/ngsi-ld/v1/types** використовуючи сформовані заголовки.

Отримуємо дані відповідь від сервера та декодуємо їх задля виведення на екран. В результаті отримаємо такий набір даних з переліком доступних списків даних:


```
{
  "id": "urn:ngsi-ld:EntityTypeList:c9b73ee2-189f-11ef-a4a6-0242ac120108",
  "type": "EntityTypeList",
  "typeList": [
    "AgriCarbonFootprint",
    "AgriCrop",
    "AgriFarm",
    "AgriGreenhouse",
    "AgriParcel",
    "AgriParcelOperation",
    "AgriParcelRecord",
    "AgriSoil",
    "AgriSoilState",
    "AgriYield",
    "Person"
  ]
}
```


*Код 3.1.3. Отриманий набір даних у форматі **JSON**.*

Отримаємо зміст списку під назвою AgriCarbonFootprint за допомогою наступного коду:

```
conn = http.client.HTTPSConnection("ds1.nosc-ua-dih.app")
```

```
payload = ""

headers = {
    'User-Agent': "insomnia/9.1.1",
    'X-Auth-token': get_oauth2_token()
}

conn.request("GET", "/ngsi-ld/v1/entities?type=AgriCarbonFootprint", payload,
headers)

res = conn.getresponse()
data = res.read()

# print(data.decode("utf-8"))

# Виведення даних у зручному для читання форматі
print(json.dumps(data.decode("utf-8"), indent=4))
```

Код 3.1.4. Процес отримання списку даних ***AgriCarbonFootprint****.*

Додамо в тіло запиту **?type=AgriCarbonFootprint** та отримаємо великий об’єм даних про вуглецеві викиди.

Нижче наведено перший елемент списку даних:


```
{
   "id": "urn:ngsi-ld:AgriCarbonFootprint:01b2a90d-3c40-4f13-8537-d2e04e5f6337",
   "type": "AgriCarbonFootprint",
   "https://smartdatamodels.org/dataModel.Agrifood/hasAgriCrop": {
     "type": "Relationship",
     "object": "urn:ngsi-ld:AgriCrop:bd4c1d19-081b-4f95-ae66-cbeab5ba28b1"
   },
   "https://smartdatamodels.org/dataModel.Agrifood/hasAgriParcel": {
     "type": "Relationship",
     "object": "urn:ngsi-ld:AgriParcel:8313680f-8155-41b2-8ccb-5d07bc39d125"
   },
   "carbonFootprint": {
     "type": "Property",
     "value": {
       "value": 5,
       "accuracyPercent": 10,
       "minValue": 1,
       "unitText": "Tons"
     }
```

```
  },
  "estimationEndAt": {
   "type": "Property",
   "value": {
    "@type": "DateTime",
    "@value": "2017-05-04T12:30:00Z"
   }
  },
  "estimationStartAt": {
   "type": "Property",
   "value": {
    "@type": "DateTime",
    "@value": "2017-05-04T12:30:00Z"
   }
  },
  "hasAgriParcelOperation": {
   "type": "Relationship",
   "object": "urn:ngsi-ld:AgriParcelOperation:17f06e2c-6fea-4d8c-ae90-316bfdaf4b32"
  },
  "hasAgriYield": {
   "type": "Relationship",
   "object": "urn:ngsi-ld:AgriYield:837ab51a-46a1-4743-8d0f-eaf79f39525c"
  }
 }
```


*Код 3.1.5. Перший елемент з отриманого набору даних у форматі **JSON**.*

3.2. Інтеграція отриманих даних з блокчейн платформою OKP4

3.2.1. Підготовка необхідних інструментів

Перш за все потрібно встановити мову програмування **Rust** та середовище розробки, я обрав **RustRover**. В майбутньому ми будемо використовувати **Rust** для написання смарт контракту.

Для цього звернемося до офіційних джерел за посиланнями:

- Встановлення **Rust**: https://www.rust-lang.org/tools/install
- Встановлення **RustRover**: https://www.jetbrains.com/rust/

Також потрібно встановити **CosmosSDK**, **WebAssembly** (**WASM**) та **OKP4 CLI** (**Command Line Interface**), тут починаються перші складнощі, оскільки я використовую **Windows**, а підтримка **OKP4 CLI** на **Windows** відсутня, доведеться встановлювати **WSL** (Windows subsystem for **Linux**), розпочнемо саме з цього кроку.

1. Встановлення WSL:

Відкриваємо Windows PowerShell від імені адміністратора та вводимо відповідну команду:

wsl –install

Очікуємо завершення завантаження, отримуємо успішний результат.

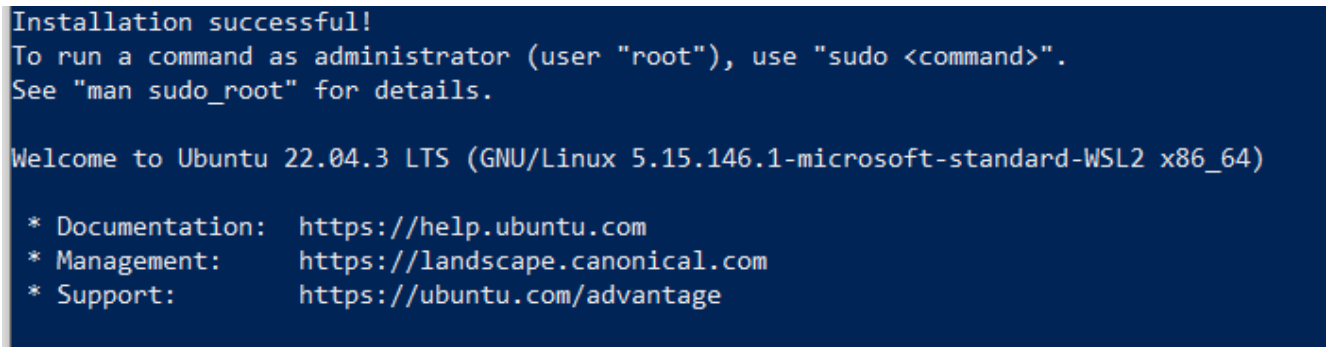

*Рис 3.2.1. Успішне встановлення **WSL**.*

2. Налаштовуємо Ubuntu:

Відкриваємо щойно встановлений застосунок. Створюємо користувача системи заповнюючи логін та пароль для входу після чого актуалізуємо залежності за допомогою наступних команд.

sudo apt update

sudo apt upgrade

sudo apt install curl tar

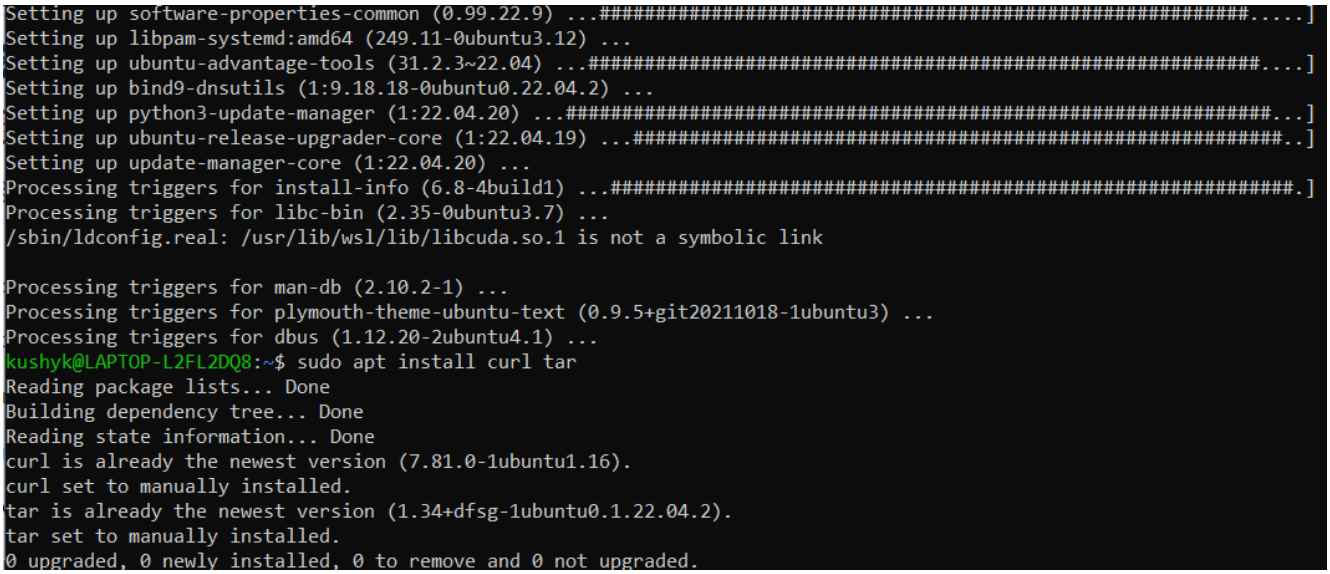

Рис 3.2.2. Частина процесу виконання наведених вище команд.

3. Завантаження та встановлення OKP4 CLI:

Виконаємо наступну команду:

curl -L "https://github.com/axone-protocol/axoned/releases/download/v8.0.0/axoned-8.0.0-linux-amd64.tar.gz" -o okp4d.tar.gz

В результаті її виконання завантажується архів з необхідними нам даними.

За допомогою команди: ***ls -l***

перевіримо чи все добре і дані збереглися без помилок

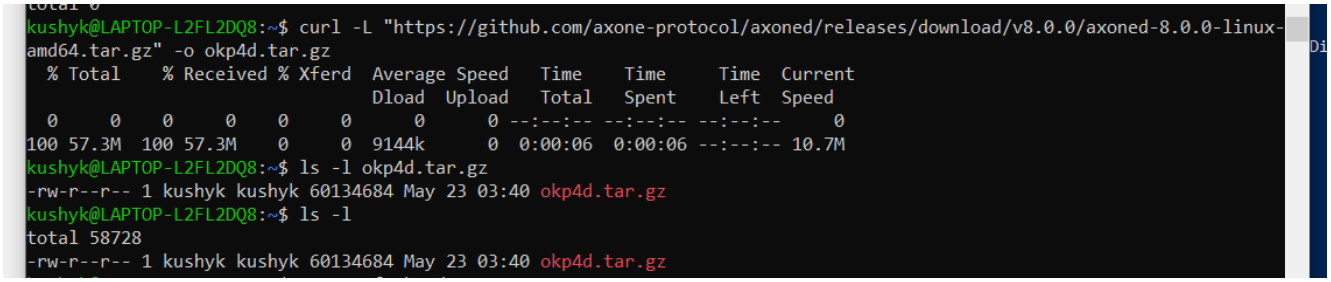

*Рис 3.2.3. Результат завантаження **OKP4 CLI**.*

Розпакуємо отриманий архів за допомогою команди:

Tar -tzf okp4d.tar.gz

Встановимо **OKP4 CLI** запустивши:

sudo mv target/release/axoned-8.0.0-linux-amd64.tar.gz /usr/local/bin/

Впевнимося в успішності встановлення використовуючи:

okp4d version

Отримуємо версію **okp4d**, що означає що встановлення відбулося успішно.

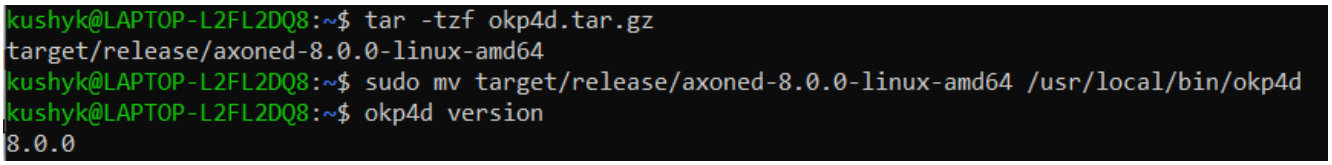

*Рис 3.2.4. Результат встановлення **OKP4 CLI**.*

4. Встановлення WebAssembly (WASM):

Для встановлення WebAssembly потрібно виконати наступну команду:

Cargo install wasm-pack

Проте, в результаті виконання команди, я зіштовхнувся з помилкою:

```
kushyk@LAPTOP-L2FL2DQ8:~$ wasm-pack --version
wasm-pack: command not found
kushyk@LAPTOP-L2FL2DQ8:~$ cargo install wasm-pack
Command 'cargo' not found, but can be installed with:
sudo snap install rustup  # version 1.27.1, or
sudo apt  install cargo   # version 0.67.1+ds0ubuntu0.libgit2-0ubuntu0.22.04.2
See 'snap info rustup' for additional versions.
```

Рис 3.2.5.Помилка виконання наведеної команди.

Для вирішення проблеми потрібно встановити **Rust** і **Cargo**, використаємо відповідні команди:

curl –proto ‘=https’ –tlsv1.2 -sSf https://sh.rustup.rs | sh

та

source $HOME/.cargo/env

```
kushyk@LAPTOP-L2FL2DQ8:~$ curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
info: downloading installer

Welcome to Rust!

This will download and install the official compiler for the Rust
programming language, and its package manager, Cargo.
```

Рис 3.2.6. Встановлення **Rust**.

Використовуючи наступну команду встановимо **Cargo**:

source $HOME/.cargo/env

В результаті успішного виконання команди повторно виконуємо команду встановлення **WASM**:

```
Rust is installed now. Great!

To get started you may need to restart your current shell.
This would reload your PATH environment variable to include
Cargo's bin directory ($HOME/.cargo/bin).

To configure your current shell, you need to source
the corresponding env file under $HOME/.cargo.

This is usually done by running one of the following (note the leading DOT):
. "$HOME/.cargo/env"            # For sh/bash/zsh/ash/dash/pdksh
source "$HOME/.cargo/env.fish"  # For fish
kushyk@LAPTOP-L2FL2DQ8:~$ source $HOME/.cargo/env
kushyk@LAPTOP-L2FL2DQ8:~$ cargo install wasm-pack
    Updating crates.io index
  Downloaded wasm-pack v0.12.1
  Downloaded 1 crate (423.6 KB) in 1.48s
  Installing wasm-pack v0.12.1
    Updating crates.io index
  Downloaded addr2line v0.21.0
  Downloaded serde_ignored v0.1.10
  Downloaded atty v0.2.14
  Downloaded adler v1.0.2
  Downloaded autocfg v1.3.0
  Downloaded tinyvec_macros v0.1.1
  Downloaded time-core v0.1.2
  Downloaded version_check v0.9.4
```

Рис 3.2.7. Встановлення **Cardgo** та **WASM**.

Проте знову стикаємося з помилками:

```
error: linker `cc` not found
  |
  = note: No such file or directory (os error 2)

error: could not compile `serde` (build script) due to 1 previous error
warning: build failed, waiting for other jobs to finish...
error: could not compile `proc-macro2` (build script) due to 1 previous error
error: could not compile `rustix` (build script) due to 1 previous error
error: could not compile `libc` (build script) due to 1 previous error
error: could not compile `serde_json` (build script) due to 1 previous error
error: could not compile `typenum` (build script) due to 1 previous error
error: failed to compile `wasm-pack v0.12.1`, intermediate artifacts can be found at `/tmp/cargo-installqtELPt`.
To reuse those artifacts with a future compilation, set the environment variable `CARGO_TARGET_DIR` to that path.
```

Рис 3.2.8. Чергові помилки при спробі встановлення **WASM**.

Задля виправлення проблеми потрібно додатково встановити **build-essential**:

sudo apt update

sudo apt install build-essential

sudo apt install pkg-config libssl-dev

Виконавши вищенаведені команди успішно завантажуємо та встановлюємо необхідні для подальшої роботи пакети.

```
/sbin/ldconfig.real: /usr/lib/wsl/lib/libcuda.so.1 is not a symbolic link

kushyk@LAPTOP-L2FL2DQ8:~$ sudo apt install pkg-config libssl-dev
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Suggested packages:
  libssl-doc
The following NEW packages will be installed:
  libssl-dev pkg-config
0 upgraded, 2 newly installed, 0 to remove and 0 not upgraded.
Need to get 2425 kB of archives.
After this operation, 12.5 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libssl-dev amd64 3.0.2-0ubuntu1.15 [2376 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy/main amd64 pkg-config amd64 0.29.2-1ubuntu3 [48.2 kB]
Fetched 2425 kB in 1s (4445 kB/s)
Selecting previously unselected package libssl-dev:amd64.
(Reading database ... 29953 files and directories currently installed.)
Preparing to unpack .../libssl-dev_3.0.2-0ubuntu1.15_amd64.deb ...
Unpacking libssl-dev:amd64 (3.0.2-0ubuntu1.15) ...
Selecting previously unselected package pkg-config.
Preparing to unpack .../pkg-config_0.29.2-1ubuntu3_amd64.deb ...
Unpacking pkg-config (0.29.2-1ubuntu3) ...
Setting up libssl-dev:amd64 (3.0.2-0ubuntu1.15) ...
Setting up pkg-config (0.29.2-1ubuntu3) ...
Processing triggers for man-db (2.10.2-1) ...
```

Рис 3.2.9. Успішне встановлення необхідних інструментів.

Вкотре намагаємося повторити команду встановлення wasm і на цей раз нас чекає успіх, додатково перевіряємо що все встановилося без помилок виконавши команду що повертає версію відповідного програмного забезпечення.

```
   Compiling zip v0.6.6
   Compiling binary-install v0.2.0
   Compiling wasm-pack v0.12.1
    Finished `release` profile [optimized] target(s) in 34.15s
  Installing /home/kushyk/.cargo/bin/wasm-pack
   Installed package `wasm-pack v0.12.1` (executable `wasm-pack`)
kushyk@LAPTOP-L2FL2DQ8:~$ wasm-pack --version
wasm-pack 0.12.1
kushyk@LAPTOP-L2FL2DQ8:~$
```

Рис 3.2.10. Успішне встановлення необхідних інструментів.

Додатково потрібно встановити цільову платформу wasm32, виконаємо необхідну команду:

rustup target add wasm32-unknown-unknown

```
kushyk@LAPTOP-L2FL2DQ8:/mnt/c/Users/minep/RustroverProjects/carbon_footprint_contract$ rustup target add wasm32-unknown-
unknown
info: downloading component 'rust-std' for 'wasm32-unknown-unknown'
info: installing component 'rust-std' for 'wasm32-unknown-unknown'
```

Рис 3.2.11. Успішне встановлення цільової платформи.

3.2.2. Використання мови програмування Rust для написання смарт контракту

Створимо **Rust** проєкт за допомогою **RustRover**:

- Відкриваємо **Rust Rover**.
- Обираємо "**Create New Project**".
- Обираємо "**Rust**" як тип проекту.
- Обираємо директорію для створення проєкту.
- Називаємо проект, наприклад: **carbon_footprint_contract**.
- Обираємо шаблон **WebAssembly**.
- Натискаємо **install cargo-generate using Cargo.**
- Створюємо проєкт натискаючи кнопку "**Create**".

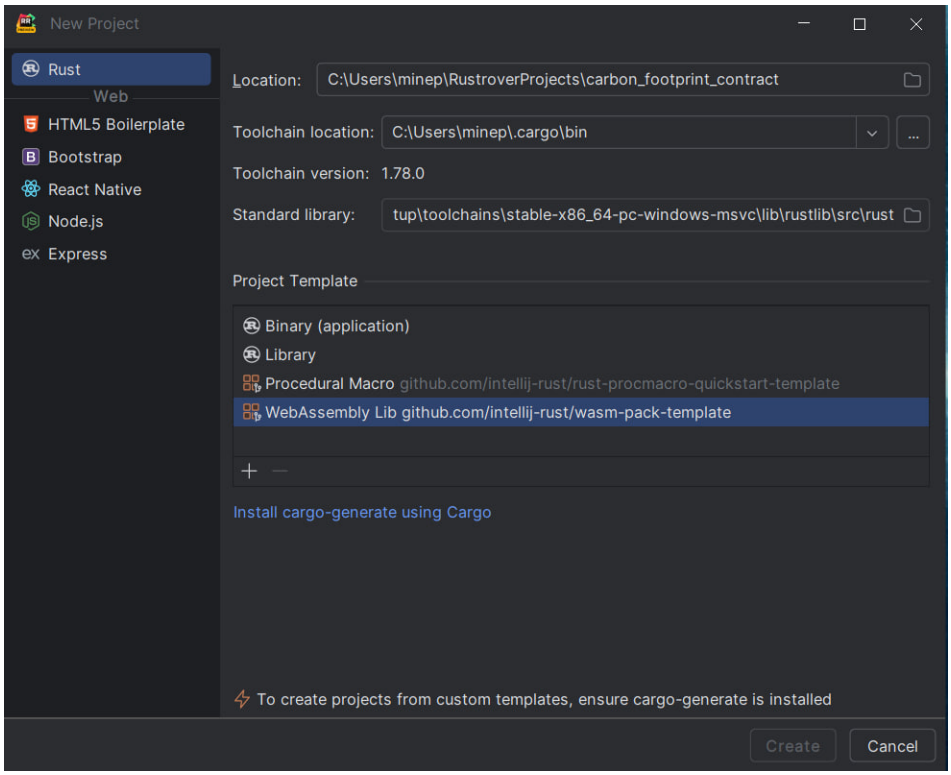

Рис 3.2.12. Інтерфейс створення проєкту.

Тепер Розберемо процес створення смарт контракту.

Перш за все створимо проєкт, після чого додамо необхідні залежності імпортуємо потрібні для роботи бібліотеки:

У файлі **Cargo.toml** додаємо:

[dependencies]
wasm-bindgen = "0.2.63"
cosmwasm-std = "2.0.3"
cosmwasm-storage = "1.5.2"
schemars = "0.8"
serde = { version = "1.0", features = ["derive"] }
bincode = "2.0.0-rc.3"
serde_json = "1.0"

*Тепер переходимо у файл **lib.rs**:*

use cosmwasm_std::{attr, Binary, Deps, DepsMut, Env, MessageInfo, Response, StdResult, Uint128};
use schemars::JsonSchema;
use serde::{Deserialize, Serialize};
use bincode::{Encode, Decode, config::standard};

Розглянемо детальніше:

- **cosmwasm_std:** Основна бібліотека для взаємодії з **CosmWasm**, що надає основні типи і функції для роботи зі смарт-контрактами.

- **schemars::JsonSchema:** Дозволяє автоматично генерувати **JSON**-схеми для ваших структур.

- **serde::{Deserialize, Serialize}:** Використовується для серіалізації та десеріалізації даних.

- **bincode:** Бібліотека для кодування і декодування даних у компактному форматі.

Тепер створимо необхідні структури даних:

```
#[derive(Serialize, Deserialize, Clone, Debug, PartialEq, JsonSchema)]
pub struct InstantiateMsg {}
```

InstantiateMsg: Порожнє повідомлення, яке використовується для ініціалізації контракту.

```
#[derive(Serialize, Deserialize, Clone, Debug, PartialEq, JsonSchema)]
pub struct ExecuteMsg {
    pub add_carbon_footprint: Option<AddCarbonFootprint>,
}
```

ExecuteMsg: Використовується для виконання різних команд контракту. У цьому випадку він має опцію **AddCarbonFootprint.**

```
#[derive(Serialize, Deserialize, Clone, Debug, PartialEq, JsonSchema, Encode, Decode)]
pub struct AddCarbonFootprint {
    pub id: String,
    pub has_agri_crop: String,
    pub has_agri_parcel: String,
    pub carbon_footprint: CarbonFootprint,
```

```
    pub estimation_start_at: String,
    pub estimation_end_at: String,
    pub has_agri_parcel_operation: String,
    pub has_agri_yield: String,
}
```

AddCarbonFootprint: Структура для представлення даних про вуглецевий слід, яка включає ідентифікатор, інформацію про агрокультуру, наявності сільськогосподарських ділянок, дані про вуглецевий слід, час початку та закінчення оцінки, інформацію про операцію та врожайність.

```
#[derive(Serialize, Deserialize, Clone, Debug, PartialEq, JsonSchema, Encode, Decode)]
pub struct CarbonFootprint {
    pub value: Uint128,
    pub accuracy_percent: u8,
    pub min_value: Uint128,
    pub unit_text: String,
}
```

CarbonFootprint: Структура для представлення специфічних даних про вуглецевий слід, включаючи значення, точність, мінімальне значення та одиницю виміру.

```
#[derive(Serialize, Deserialize, Clone, Debug, PartialEq, JsonSchema)]
pub struct QueryMsg {
    pub get_carbon_footprint: Option<String>,
}
```

QueryMsg: Використовується для запитів до контракту. У цьому випадку запитуємо вуглецевий слід за певним ідентифікатором. Це дозволить в майбутньому отримати дані про вуглецевий слід певної ферми прямо з блокчейну.

```
#[derive(Serialize, Deserialize, Clone, Debug, PartialEq, JsonSchema)]
pub struct CarbonFootprintResponse {
    pub id: String,
    pub has_agri_crop: String,
    pub has_agri_parcel: String,
    pub carbon_footprint: CarbonFootprint,
    pub estimation_start_at: String,
    pub estimation_end_at: String,
    pub has_agri_parcel_operation: String,
    pub has_agri_yield: String, }
```

CarbonFootprintResponse: Структура відповіді на запит вуглецевого сліду.

```
pub fn instantiate(
    deps: DepsMut,
    _env: Env,
    _info: MessageInfo,
    _msg: InstantiateMsg,
) -> StdResult<Response> {
    Ok(Response::default())
}
```

instantiate: Функція ініціалізації контракту. У цьому випадку вона не виконує жодних дій і просто повертає стандартну відповідь.

```
pub fn execute(
```

```
    deps: DepsMut,
    _env: Env,
    _info: MessageInfo,
    msg: ExecuteMsg,
) -> StdResult<Response> {
    match msg.add_carbon_footprint {
        Some(carbon_footprint) => try_add_carbon_footprint(deps, carbon_footprint),
        None => Err(cosmwasm_std::StdError::generic_err("Invalid ExecuteMsg")),
    }
}
```

execute: Основна функція виконання команд. Вона перевіряє, чи містить повідомлення **add_carbon_footprint**, і викликає відповідну функцію.

```
pub fn try_add_carbon_footprint(
    deps: DepsMut,
    carbon_footprint: AddCarbonFootprint,
) -> StdResult<Response> {
    let key = carbon_footprint.id.as_bytes();
    let value = bincode::encode_to_vec(&carbon_footprint, standard()).unwrap();
    deps.storage.set(key, &value);

    Ok(Response::new().add_attributes(vec![
        attr("action", "add_carbon_footprint"),
        attr("id", carbon_footprint.id),
    ]))
}
```

try_add_carbon_footprint: Функція, яка додає новий вуглецевий слід до сховища. Вона використовує **bincode** для серіалізації даних і зберігає їх за ідентифікатором.

```
pub fn query(deps: Deps, _env: Env, msg: QueryMsg) -> StdResult<Binary> {
    match msg.get_carbon_footprint {
        Some(id) => query_carbon_footprint(deps, id),
        None => Err(cosmwasm_std::StdError::generic_err("Invalid QueryMsg")),
    }
}
```

query: Основна функція запиту. Вона перевіряє, чи містить повідомлення **get_carbon_footprint**, і викликає відповідну функцію.

```
fn query_carbon_footprint(deps: Deps, id: String) -> StdResult<Binary> {
    let key = id.as_bytes();
    let value = deps.storage.get(key);

    match value {
        Some(value) => {
            let (carbon_footprint, _): (AddCarbonFootprint, usize) = bincode::decode_from_slice(&value, standard()).unwrap();
            let response = CarbonFootprintResponse {
                id: carbon_footprint.id,
                has_agri_crop: carbon_footprint.has_agri_crop,
                has_agri_parcel: carbon_footprint.has_agri_parcel,
                carbon_footprint: carbon_footprint.carbon_footprint,
                estimation_start_at: carbon_footprint.estimation_start_at,
                estimation_end_at: carbon_footprint.estimation_end_at,
```

```
                has_agri_parcel_operation:
carbon_footprint.has_agri_parcel_operation,
                has_agri_yield: carbon_footprint.has_agri_yield,
            };
            to_binary(&response)
        }
        None => Err(cosmwasm_std::StdError::not_found("CarbonFootprint")),
    }
}
```

query_carbon_footprint: Функція, яка виконує запит вуглецевого сліду за заданим ідентифікатором. Вона використовує **bincode** для десеріалізації даних зі сховища.

```
fn to_binary<T: ?Sized>(value: &T) -> StdResult<Binary>
where
    T: serde::Serialize,
{
    let binary = Binary::from(serde_json::to_vec(value)?);
    Ok(binary)
}
```

to_binary: Утилітна функція для конвертації серіалізованого об'єкта до формату **Binary**, що використовується в **CosmWasm**.

Отже ми створили смарт контракт, що складається з ініціалізацію контракту, додавання нових записів про вуглецеві сліди та запити для отримання цих записів.

В результаті ми створили смарт контракт, який дозволяє додавати дані про вуглецевий слід у блокчейн **AXONE**, а також дізнаватися необхідну інформацію з

блокчейну отримуючи конкретні дані про вуглецеві викиди використовуючи **id**. Ми використовуємо серіалізацію і десеріалізацію для зберігання і отримання даних з використанням **bincode** та **serde**.

3.2.3. Додавання даних у блокчейн AXONE

Спробуємо скомпілювати наш проєкт за допомогою команди:

cargo build --target wasm32-unknown-unknown –release

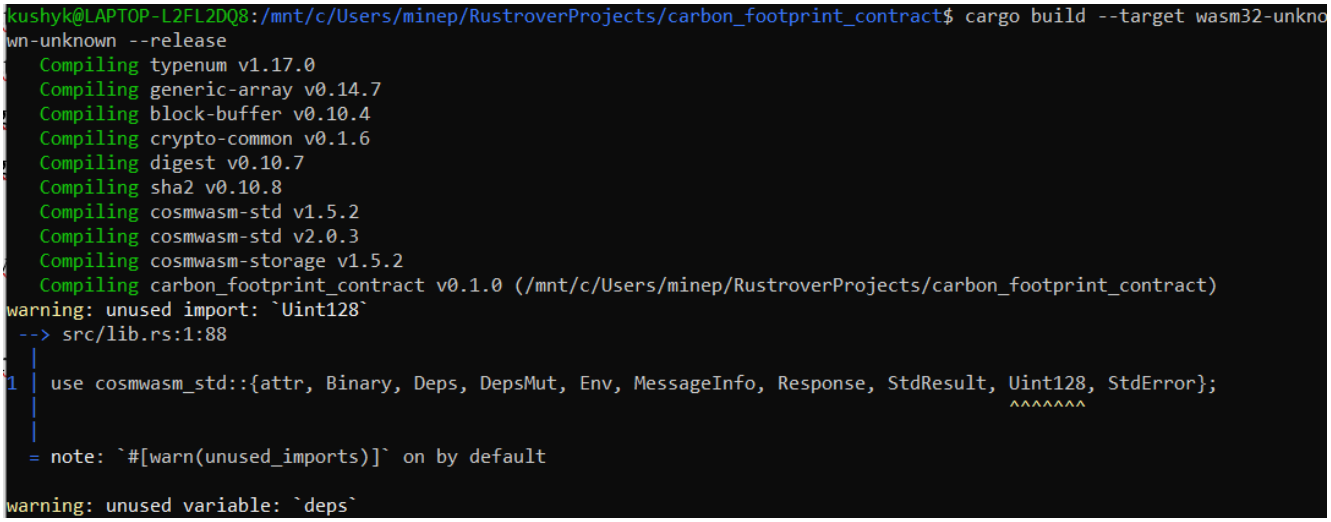

Рис 3.2.13. Компіляція проєкту.

Переходимо у директорію де лежить скомпільований файл:

cd target/wasm32-unknown-unknown/release/

Надалі цей контракт можна використати для передачі даних у відповідному форматі в мережу, проте для цього потрібно синхронізувати власну ноду зі всіма створеними вузлами, цей процес займає 2-3 доби в залежності від мережі яку використовувати, у зв’язку з неможливістю утримання стабільного зв’язку з мережею протягом такого довгого періоду, в наслідок аварійних відключень, виконаємо задачу іншим, дещо простішим шляхом.

Задля виконання задачі скористаємося послугами **OKP4 Objectarium [19]**

OKP4 Objectarium – це компонент інфраструктури **OKP4** мережі, яка забезпечує зберігання та управління об'єктами (даними) у вигляді **JSON**. Основна функція **Objectarium** – це надання децентралізованого рішення для зберігання даних, яке підтримує різноманітні формати і типи інформації.

Оскільки транзакцію потрібно оплатити, перш за все створимо гаманець у відповідній мережі:

okp4d keys add "wallet-name" --keyring-backend test

```
kushyk@LAPTOP-L2FL2DQ8:~$ okp4d keys add kmaTop --keyring-backend test

- address: okp4185vcxfhxg7src6dz9x6gvwhjltes5ptmt6fnw6
  name: kmaTop
  pubkey: '{"@type":"/cosmos.crypto.secp256k1.PubKey","key":"AxxE70wWEVZp+97+nzdGWDeYNlQfvkyMBdSnqG4JkRBi"}'
  type: local


**Important** write this mnemonic phrase in a safe place.
It is the only way to recover your account if you ever forget your password
```

Рис 3.2.14. Створення гаманця.

Зберігаємо дані у полі **address**.

Можна поповнити гаманець за допомогою безкоштовних тестових токенів, для цього перейдемо за посиланням https://faucet.okp4.network/

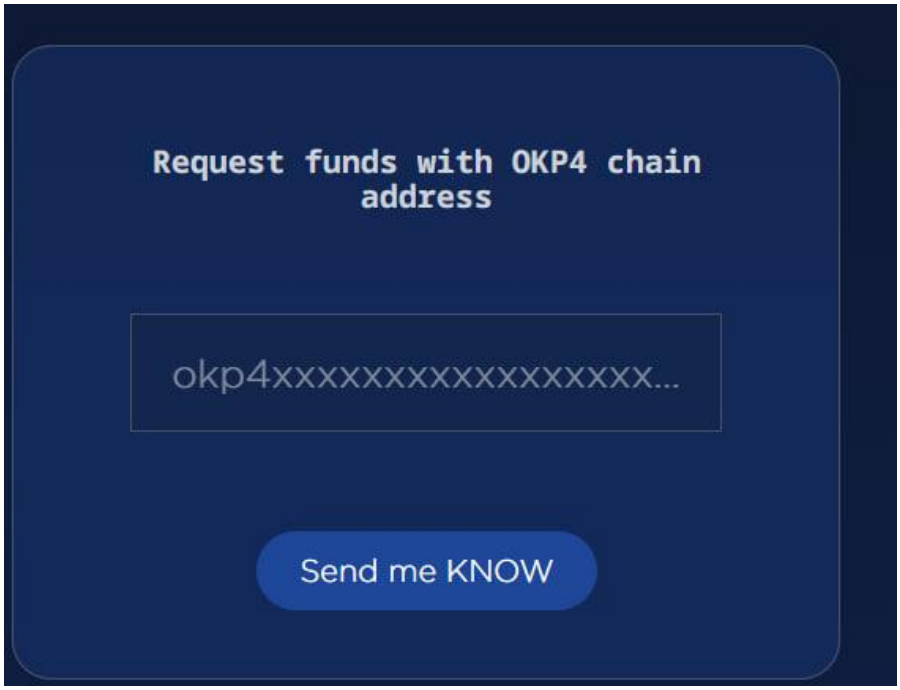

Рис 3.2.15. Поле для введення адреси нашого гаманця.

Перевіряємо чи зарахувалися токени:

okp4d query bank balances okp4185vcxfhxg7src6dz9x6gvwhjltes5ptmt6fnw6 --node https://api.drunemeton.okp4.network/rpc

```
kushyk@LAPTOP-L2FL2DQ8:~$ okp4d query bank balances okp4185vcxfhxg7src6dz9x6gvwhjltes5ptmt6fnw6 --node https://api.drun
meton.okp4.network/rpc
balances:
- amount: "1000000"
  denom: uknow
pagination:
  total: "1"
```

Рис 3.2.15. Баланс гаманця.

Тепер можна створювати екземпляр **Objectarium**:

*okp4d tx wasm instantiate 50 *

*> --label "my-storage" *

*> --from okp4185vcxfhxg7src6dz9x6gvwhjltes5ptmt6fnw6 *

*> --admin okp4185vcxfhxg7src6dz9x6gvwhjltes5ptmt6fnw6 *

*> --gas 1000000 *

*> --chain-id "okp4-drunemeton-1" *

*> --node https://api.drunemeton.okp4.network/rpc *

*> --keyring-backend test *

> '{"bucket":"my-bucket"}'

```
kushyk@LAPTOP-L2FL2DQ8:~$ okp4d tx wasm instantiate 50 \
> --label "my-storage" \
> --from okp4185vcxfhxg7src6dz9x6gvwhjltes5ptmt6fnw6 \
dmin okp> --admin okp4185vcxfhxg7src6dz9x6gvwhjltes5ptmt6fnw6 \
> --gas 1000000 \
> --chain-id "okp4-drunemeton-1" \
> --node https://api.drunemeton.okp4.network/rpc \
> --keyring-backend test \
> '{"bucket":"my-bucket"}'
auth_info:
  fee:
    amount: []
    gas_limit: "1000000"
    granter: ""
    payer: ""
  signer_infos: []
  tip: null
```

Рис 3.2.15. Успішно створений екземпляр.

Знаходимо поле **txhash** та зберігаємо його, воно знадобиться нам для отримання адреси смарт контракту.

```
signatures: []
confirm transaction before signing and broadcasting [y/N]: y
code: 0
codespace: ""
data: ""
events: []
gas_used: "0"
gas_wanted: "0"
height: "0"
info: ""
logs: []
raw_log: ""
timestamp: ""
tx: null
txhash: 1114722D2696E64C80C9764BDF6A5D233038837AFD7AC888E49FABDFF639B899
```

Рис 3.2.16. Зміст поля **txhash**.

Отримуємо адресу смарт контракту:

okp4d query tx
1114722D2696E64C80C9764BDF6A5D233038837AFD7AC888E49FABDFF639B899
**

*--node https://api.drunemeton.okp4.network/rpc *

*--output json *

| jq '.events[] | select(.type == "instantiate").attributes[] | select(.key == "_contract_address").value'

```
kushyk@LAPTOP-L2FL2DQ8:~$ okp4d query tx 1114722D2696E64C80C9764BDF6A5D233038837AFD7AC888E49FABDFF639B899 \
--node h> --node https://api.drunemeton.okp4.network/rpc \
> --output json \
> | jq '.events[] | select(.type == "instantiate").attributes[] | select(.key == "_contract_address").value'
"okp410emt4hxmeyr8mjxayyt8huelzd7fpntmly8vus5puelqde6kn8xqsednk9"
```

Рис 3.2.17. Зміст поля **txhash**.

Зберігаємо виділені зеленим символи.

Тепер нам потрібно створити файл, в який ми помістимо дані, що плануємо завантажити у блокчейн, а саме **json** об’єкт, що містить інформацію про вуглецеві викиди.

echo "json data" > content.txt

```
kushyk@LAPTOP-L2FL2DQ8:~$ echo "[{\"id\":\"urn:ngsi-ld:AgriCarbonFootprint:01b2a90d-3c40-4f13-8537-d2e04e5f6337\",\"type
\":\"AgriCarbonFootprint\",\"https://smartdatamodels.org/dataModel.Agrifood/hasAgriCrop\":{\"type\":\"Relationship\",\"o
bject\":\"urn:ngsi-ld:AgriCrop:bd4c1d19-081b-4f95-ae66-cbeab5ba28b1\"},\"https://smartdatamodels.org/dataModel.Agrifood/
hasAgriParcel\":{\"type\":\"Relationship\",\"object\":\"urn:ngsi-ld:AgriParcel:8313680f-8155-41b2-8ccb-5d07bc39d125\"},\
"carbonFootprint\":{\"type\":\"Property\",\"value\":{\"value\":5,\"accuracyPercent\":10,\"minValue\":1,\"unitText\":\"To
ns\"}},\"estimationEndAt\":{\"type\":\"Property\",\"value\":{\"@type\":\"DateTime\",\"@value\":\"2017-05-04T12:30:00Z\"}
},\"estimationStartAt\":{\"type\":\"Property\",\"value\":{\"@type\":\"DateTime\",\"@value\":\"2017-05-04T12:30:00Z\"}},\
"hasAgriParcelOperation\":{\"type\":\"Relationship\",\"object\":\"urn:ngsi-ld:AgriParcelOperation:17f06e2c-6fea-4d8c-ae9
0-316bfdaf4b32\"},\"hasAgriYield\":{\"type\":\"Relationship\",\"object\":\"urn:ngsi-ld:AgriYield:837ab51a-46a1-4743-8d0f
-eaf79f39525c\"}},{\"id\":\"urn:ngsi-ld:AgriCarbonFootprint:04d01771-a612-47f5-aa68-b0e25bb32ab9\",\"type\":\"AgriCarbon
Footprint\",\"https://smartdatamodels.org/dataModel.Agrifood/hasAgriCrop\":{\"type\":\"Relationship\",\"object\":\"urn:n
gsi-ld:AgriCrop:bd4c1d19-081b-4f95-ae66-cbeab5ba28b1\"},\"https://smartdatamodels.org/dataModel.Agrifood/hasAgriParcel\"
:{\"type\":\"Relationship\",\"object\":\"urn:ngsi-ld:AgriParcel:8313680f-8155-41b2-8ccb-5d07bc39d125\"},\"carbonFootprin
t\":{\"type\":\"Property\",\"value\":{\"value\":5,\"accuracyPercent\":10,\"minValue\":1,\"unitText\":\"Tons\"}},\"estima
tionEndAt\":{\"type\":\"Property\",\"value\":{\"@type\":\"DateTime\",\"@value\":\"2017-05-04T12:30:00Z\"}},\"estimationS
tartAt\":{\"type\":\"Property\",\"value\":{\"@type\":\"DateTime\",\"@value\":\"2017-05-04T12:30:00Z\"}},\"hasAgriParcelO
peration\":{\"type\":\"Relationship\",\"object\":\"urn:ngsi-ld:AgriParcelOperation:436844d6-858c-48a3-8de3-1172e287ff71\
```

Рис 3.2.18. Збереження **JSON** у файл, для подальшого завантаження у мережу.

```
kushyk@LAPTOP-L2FL2DQ8:~$ ls
content.txt  go  go1.22.3.linux-amd64.tar.gz
```

Рис 3.2.19. Впевнимося що файл успішно створено.

Намагаємося виконати смарт контракт:

okp4d tx wasm execute
okp410emt4hxmeyr8mjxayyt8huelzd7fpntmly8vus5puelqde6kn8xqsednk9
--from okp4185vcxfhxg7src6dz9x6gvwhjltes5ptmt6fnw6
*--gas 1000000 *
*--chain-id okp4-drunemeton-1 *
*--node https://api.drunemeton.okp4.network/rpc *
*--keyring-backend test *
"{"store_object":{"data": "$(cat content.txt | base64 | tr -d '\n\r')", "pin":true}}"

Рис 3.2.20. Спроба виконання смарт контракту.

Проте отримуємо помилку, ми передаємо досить великий об’єм даних, так що потрібно збільшити **gas limit:**

Рис 3.2.21. Спроба виконання смарт контракту з підвищеним **gas limit**.

Транзакція пройшла успішно, перевіримо це отримавши **id** створеного об’єкту використавши:

okp4d query tx BAA8EF2A54C81EE44CBB598D35A4835DF334C889971652AF7337A72F68028C5A --node https://api.drunemeton.okp4.network/rpc

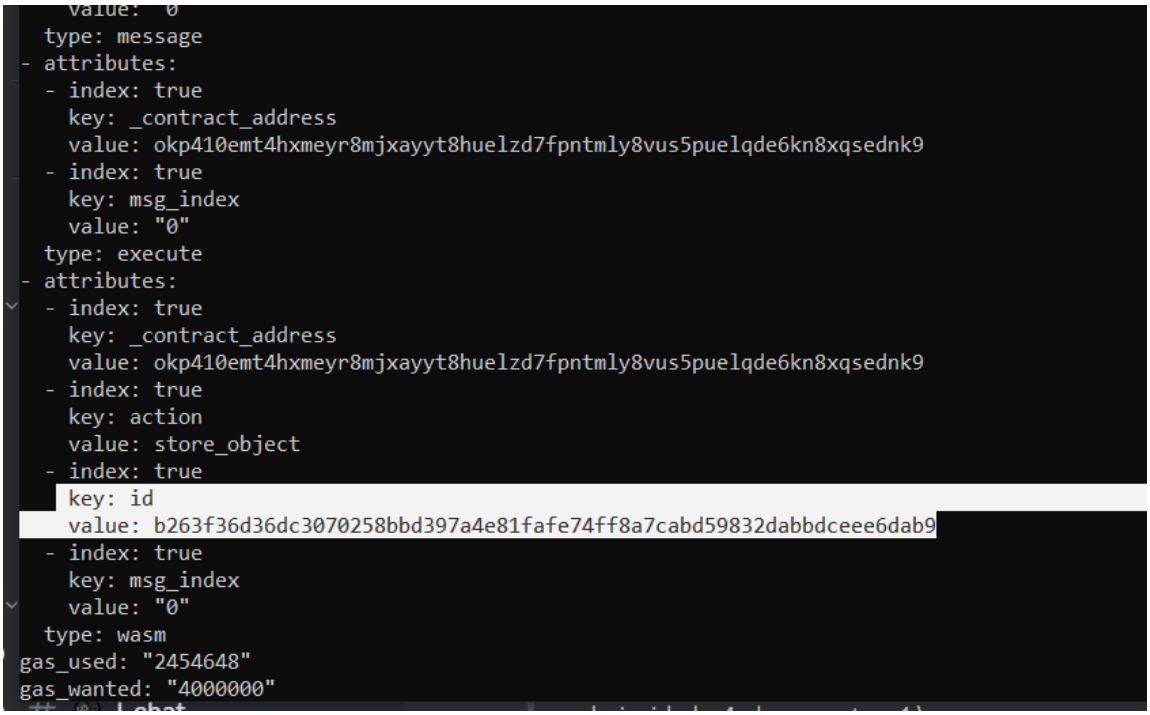

*Рис 3.2.22. Знаходимо поле **key id** і зберігаємо значення.*

Тепер отримаємо і розшифруємо дані з блокчейну, щоб остаточно впевнитися в успішності виконаних дій:

*okp4d query wasm contract-state smart okp410emt4hxmeyr8mjxayyt8huelzd7fpntmly8vus5puelqde6kn8xqsednk9 *

*--node h> --node https://api.drunemeton.okp4.network/rpc *

*> --output json *

*> '{"object_data": {"id": "b263f36d36dc3070258bbd397a4e81fafe74ff8a7cabd59832dabbdceee6dab9"}}' *

jq -r > | jq -r ".data" | base64 --decode

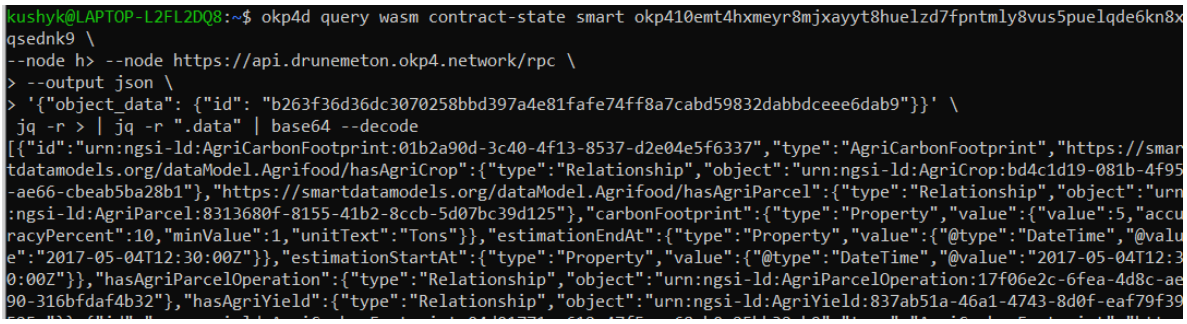

*Рис 3.2.23. Отримуємо **JSON** об’єкт, що був успішно збережений в блокчейні.*

В результаті проведених дій ми створили та поповнили гаманець, під’єдналися до мережі OKP4, зберегли необхідні дані, зберегли їх у блокчейн використавши смарт контракт, та впевнилися в успішно дій, отримавши відповідні дані з мережі.

Висновки

У даній дипломній роботі було досліджено та реалізовано інтеграцію Smart Data Space з блокчейн-платформою OKP4 для монетизації вуглецевих викидів.

В результаті теоретичної частини проведеного дослідження було розглянуто наступні теми:

- **Актуальність проблеми глобального потепління:** Робота підтверджує, що глобальне потепління є серйозною проблемою, зокрема через значний вплив вуглецевих викидів. Сільськогосподарський сектор робить вагомий внесок у цей процес, тому контроль та моніторинг викидів у цій сфері є важливими для зменшення негативного впливу на довкілля.

- **Математична модель парникового ефекту:** В одному з розділів було розглянуто чинники парникового ефекту, вплив людської діяльності на це явище, та математичне обґрунтування причин існування парникового ефекту і перспектив зміни клімату внаслідок його впливу на середню температуру Землі.

- **Важливість технологій блокчейну та Smart Data Space:** Технології блокчейну, зокрема платформа OKP4, дозволяють забезпечити прозорість та надійність даних про вуглецеві викиди, що є ключовим фактором для побудови довіри та ефективного управління цими даними. Використання Smart Data Space допомагає структурувати та стандартизувати дані, що сприяє їхньому кращому аналізу та використанню.

- **Принципи роботи технологій блокчейну та смарт контрактів:** Блокчейн-технологія забезпечує розподілений реєстр, де дані зберігаються на багатьох комп'ютерах одночасно, що гарантує їхню безпеку та незмінність. Смарт-контракти, які є програмами, що автоматично виконують умови угоди при настанні певних обставин, забезпечують автоматизацію процесів, високу

надійність і прозорість угод. Це значно знижує витрати на посередників і підвищує економічну ефективність.

- **Можливості монетизації вуглецевих викидів:** Вуглецеві кредити, отримані за допомогою інтегрованої системи, можуть бути використані для фінансування екологічних проектів, що сприяють зменшенню викидів. Це стимулює підприємства впроваджувати заходи для зменшення свого вуглецевого сліду.

Практична частина дослідження включала в себе:

- Використання сучасних технологій авторизації та взаємодії з віддаленими сховищами даних, а саме мову програмування **Python**, бібліотек **requests**, **http.client**, **JSON** та сервером управління ідентифікацією та доступом **Keyrock.**

- Роботу з командним рядком задля встановлення необхідних інструментів розробки і взаємодії з блокчейном.

- Розгляд та практичне застосування блокчейн технології **AXONE (OKP4)** і **Cosmo SDK** задля створення транзакції, що містить в собі дані про вуглецеві викиди.

- Написання смарт контракту з використанням мови програмування **Rust** та середовища розробки **RustRover.**

- Створення та відправлення блокчейн транзакції з необхідними даними у блокчейн платформу OKP4.

У результаті практичної реалізації інтеграції **Smart Data Space** з блокчейн-платформою **OKP4** було підтверджено зручність та ефективність використання

блокчейн технологій для моніторингу та управління вуглецевими викидами та їхню перспективність, що відкриває нові можливості для сталого розвитку та боротьби зі зміною клімату.

Список літератури

1. Understanding carbon footprinting for cereals and oilseeds [Електронний ресурс] – Режим доступу до ресурсу: https://projectblue.blob.core.windows.net/media/Default/Imported%20Publication%20Docs/Understanding%20carbon%20footprinting%20for%20cereals%20and%20oilseeds.pdf

2. Grape Wine Cultivation Carbon Footprint: Embracing a Life Cycle Approach across Climatic Zones [Електронний ресурс] – Режим доступу до ресурсу: https://www.mdpi.com/2077-0472/13/2/303

3. Mathematical Model for CO2 Emissions Reduction to Slow and Reverse Global Warming [Електронний ресурс] – Режим доступу до ресурсу: https://www.intechopen.com/chapters/68928

4. Introductory Chapter: Climates, Change, and Climate Change [Електронний ресурс] – Режим доступу до ресурсу: https://www.intechopen.com/chapters/71858

5. Mathematical and Statistical Modeling of Global Warming [Електронний ресурс] – Режим доступу до ресурсу: https://link.springer.com/referenceworkentry/10.1007/978-3-642-04898-2_352

6. A Mathematical Model to Investigate the Frequent Impact of Global Warming on Coastal Lives [Електронний ресурс] – Режим доступу до ресурсу: https://www.ieomsociety.org/imeom/140.pdf

7. There is unequivocal evidence that Earth is warming at an unprecedented rate. Human activity is the principal cause. [Електронний ресурс] – Режим доступу до ресурсу: https://science.nasa.gov/climate-change/evidence/

8. The Basics of Climate Change [Електронний ресурс] – Режим доступу до ресурсу: https://royalsociety.org/news-resources/projects/climate-change-evidence-causes/basics-of-climate-change/

9. Climate Science Special Report [Електронний ресурс] – Режим доступу до ресурсу: https://science2017.globalchange.gov/

10. Energy Recirculation and a Layered Radiative Atmosphere Model [Електронний ресурс] – Режим доступу до ресурсу: https://climatepuzzles.org/2021/04/recirculation-layered-radiative-atmosphere/

11. Smart Data Models [Електронний ресурс] – Режим доступу до ресурсу: https://smartdatamodels.org/

12. INTRODUCTION TO Smart Data Models Program [Електронний ресурс] – Режим доступу до ресурсу: https://docs.google.com/presentation/d/1k021ZzRyk2PtikubOFNa3KyXeXL5nTmWWxyhs0Y0r1k/edit#slide=id.g10849e1f66c_0_499

13. Fundamentals of Axone [Електронний ресурс] – Режим доступу до ресурсу: https://docs.okp4.network/technical-documentation/overview

14. Axone [Електронний ресурс] – Режим доступу до ресурсу: https://axone.xyz/

15. OKP4 та команда проєкту [Електронний ресурс] – Режим доступу до ресурсу: https://shorturl.at/anO17

16. OKP4 Обмін знаннями через онтологічно керований всесвіт даних: новий підхід до опису знань [Електронний ресурс] – Режим доступу до ресурсу: http://surl.li/tqzie

17. How Does the Blockchain Work? [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/how-does-the-blockchain-work/

18. Smart Contracts in Blockchain [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/smart-contracts-in-blockchain

19. okp4-objectarium [Електронний ресурс] – Режим доступу до ресурсу: https://docs.axone.xyz/contracts/okp4-objectarium

20. SHA-256 [Електронний ресурс] – Режим доступу до ресурсу: https://helix.stormhub.org/papers/SHA-256.pdf

21. The Mathematics of Bitcoin — SHA-256 [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/swlh/the-mathematics-of-bitcoin-74ebf6cefbb0