< CONTAINER MONITORING SYSTEM >

WWW.PARADIGMA.CL
INFO@PARADIGMA.CL

OCTOBER 21, 2018

Contenido

1. STATE OF THE ART 3
2.5 Distributed Ledger Tecnnologies 3
2.5.1 Blockchain Elements 4
2.5.1.1 Consensus Protocols 4
2.5.1.1.1 Proof-of-Work (PoW) 4
2.5.1.1.2 Proof-of-Stake (PoS) 4
2.5.1.1.3 Proof-of-Authority (PoA) 5
2.5.1.1.4 Proof-of-Elapsed Time (PoET) 5
2.5.1.1.5 Hybrid approaches 5
2.5.1.2 Smart Contracts 5
2.5.1.3 Permission Model 8
2.5.1.3.1 Permissionless (Open Ledgers) 8
2.5.1.3.2 Permissioned Ledgers 8
2.5.2 Noteworthy Blockchain Implementations 8
2.5.2.1 Public or open DLTs 8
2.5.2.1.1 Bitcoin 8
2.5.2.1.2 Ethereum 8
2.5.2.1.3 Cardano 9
2.5.2.1.4 IOTA 9
2.5.2.2 Private/Permissioned DLTs 10
2.5.2.2.1 Hyperledger Fabric 10
2.5.2.2.2 MultiChain 10
2.5.2.2.3 Corda 11
2.5.2.3 Two DLTs considered for the project 12
2.5.2.3.1 Blockstack 12
2.5.2.3.2 Kadena 14
2.5.3 Other Blockchains 15
2.5.3.1 Cosmos 15
2.5.3.2 EOS 17
2.5.3.3 Comparing some platforms 18
2.6 PACT (Language for Smart Contract) 18
2.7 The future of Blockchain 19
2.7 Storage 21
2.7.1 Cloud 21
2.7.2 CEPH 21
2.7.3 MongoDB 22
2.8 Cyberphysical Systems 23
2.9 Virtualization 23
2.9.1 Docker 24
2.9.2 LXD 24
2.10 Existing solutions for container tracking systems 25
2.10.1 Remote Containers Management by Maerks Line 25
2.10.2 Smart Container by Loginno 26
2.10.3 Smart Container by Traxens 26
2.10.4 Cellocator's Cellotrack 28
2.10.5 Triton by GlobeArea 28
2. ARCHITECTURE 28
3.1 General System Architecture 29
3.2 System Description 29
3.2.1 Solution LPWAN with star topology with LoRaWan 29
3.2.2 Solution 6loWPAN with mesh topology with WI-SUN 30

3.2.3 The IoT devices and the TMS central station 32
3.2.4 TMS central station and the TMS Node API 34
3.2.3 Blockchain with Blockstack 35
3.2.5 Smart Contract with PACT 37
3.2.6 dApp 38
3.2.7 Storage of JSON files in the Cloud and Blockchain 40
3.2.8 Dashboard 42
3.2.9 Languages 45
3.2.10 Smart Contract Mantenaince 46
3.2.11 Reports and Data Analisys 47
3.2.10 Data Model 48
4. GLOSSARY OF TÉRMS 60
5. BIBLIOGRAPHY 61

1. STATE OF THE ART

2.5 Distributed Ledger Tecnnologies

This document provides an overview of the State of the Art in blockchains and Distributed Ledger Technologies (DLTs) more generally, with the intent of employing them to support trust and realize automatic operations in the Internet of Things (IoT) through the use of Smart Contracts (realized over blockchains). For this reason, IoT characteristics and architectures are also briefly reviewed and some IoT systems that have embraced DLTs are discussed, together with a presentation of security and privacy aspects. Distributed Ledger Technologies (DLTs) provide a tamper-proof database where trust is supported in a distributed manner across a set of computers, rather than centrally through a single or group of institutions. Blockchains are one type of distributed ledgers, where data records are grouped into blocks that are linked through cryptographic hashes, thus forming a chain of blocks. The main elements of a blockchain are the consensus mechanism and the programming language; the latter can range from simple scripting to more powerful smart contract programming languages that support Turing-complete computation, can maintain a state, and support the interaction with other contracts. Blockchain systems can belong to two broad categories that differ in the policy that defines which nodes can participate in the blockchain's distributed network and the roles that they can perform. The consensus mechanism is central in providing the guarantees and properties of the DLT and is still a topic of active research. Noteworthy blockchain systems include Bitcoin, Ethereum, Cardano, IOTA, which are public (open) blockchains, and Hyperledger Fabric, MultiChain, and Corda which are permissioned (private) blockchains. Special cases for the project are Blockstack and Kadena.

Distributed Ledger Technologies (DLT) are a relatively new set of technologies, with no more than a decade of history. Despite their recent conception, DLTs have attracted tremendous attention in a number of diverse fields, generally all industries for which provable trust, accountability, and/or transparency play key business roles. Some people claim that DLTs will bring to asset management as big of a revolution as the Internet brought to communication.

In a nutshell, a DLT is a giant append-only log file replicated across a set of participating nodes. When a new log entry is to be appended, participating nodes vote on whether it complies with the DLT's rules, and come to an agreement regarding the admission and the order of new log entries. This agreement is known as consensus, and the protocol ensuring it is called the consensus protocol (of the particular DLT).

What makes DLTs a disruptive technology is that they offer, for the first time ever, a tamperproof database where trust emerges through the collaboration of a set of computers, rather than through an institution or organisation that imposes trust from the external world onto the system.

Blockchains, on the other hand, are just one type of distributed ledgers. They have become so hyped that many people perceive the two terms as synonyms. Although blockchains are distributed ledgers, not all distributed ledgers are based on blockchain technology.

In the case of blockchains, data records (log entries) are grouped into blocks. The very first block, known as the genesis block, is a special block known to everyone. Each other block links to its previous block by incorporating a cryptographic hash of the previous block's contents, creating a chain of blocks. In case a block gets tampered with, the hash stored in its successor will not match its (updated) content anymore, effectively breaking the link. That is, tampering with a block deprives it of its successors, rendering the altered block as the last block of this (altered) blockchain. Given the blockchain policy that given two blockchains that have the same genesis block and have different lengths, the longest one is considered valid and any shorter ones as orphans, the tampered with shorter blockchain will be instantly ignored by all correctly behaving nodes, rendering tampering with the block meaningless. As creating new blocks is made deliberately hard (see below), an attacker wanting to tamper with a block will have a very hard time in trying to produce a longer chain than the so-far longest chain.

2.5.1 Blockchain Elements

Each blockchain is designed as a combination of certain elements. Identifying and understanding these elements is essential to the comprehension of blockchains and their innerworkings. The main elements determining the operation of a blockchain are the following:

- Consensus Protocol
- Smart Contracts (if any)
- Permission Model

2.5.1.1 Consensus Protocols

DLTs' most innovative breakthrough is the creation of trust based on a large number of generally untrusted nodes. This is achieved through sophisticated consensus mechanisms, which, as outlined above, are central to the operation of DLTs. A number of DLT consensus mechanisms have been devised, having significant differences, yet a common goal: Enabling the entire network to decide unanimously and inadvertently on which records to include next, and in which order, into the DLT. The protocol constitutes, essentially, a voting mechanism used for filtering and ordering the records that are stored into the DLT. In the following subsections we review the most common consensus mechanisms.

2.5.1.1.1 Proof-of-Work (PoW)

Proof-of-Work (PoW) or Nakamoto consensus, is the first and most popular consensus mechanism for DLTs to date, primarily known through its use in Bitcoin. It is based on the combination of (i) the "longest chain wins" principle, (ii) a computationally hard problem to build (or mine) the next block, and (iii) a reward for mining a new block into the chain. The most often used computationally hard problem is reverse hashing, having an inherently only brute force (like) solution. Unfortunately, reverse hashing has no real-world scientific or societal causes or benefits, beyond just achieving the consensus per se. Its sole purpose is to incentivize participating nodes (known as miners) to devote their resources to the chain that has grown to be the longest, i.e., the winning chain, thereby effectively achieving a universal, decentralized consensus on which specific chain version is valid. Unless a single entity controls more than 50% of the world's mining capacity, it is in each miner's interest to abide by the rules. Unfortunately, with the increasing popularity of massive mining pools, the scenario of aggregating more than half of the world's hash power under a single entity is no longer unlikely. Thus, the danger of what is known as the 51% attack cannot be entirely ruled out. PoW's main drawback is, however, the exorbitant amount of energy it requires. For example, Bitcoin's mining power is expected to surpass the entire power consumption of countries like Ireland or Denmark by 2020. This puts the long-term viability of PoW-based consensus at stake, notably in terms of their carbon footprint and their effect on global ecology. It has, thus, fueled strong research efforts in devising alternative consensus mechanisms, some of which are described below.

2.5.1.1.2 Proof-of-Stake (PoS)

Proof-of-Stake (PoS) is an alternative consensus mechanism, addressing the two main deficiencies of PoW, namely the huge waste of energy and the 51% attack. To get the terminology straight, in PoS terms miners are called validators and the act of mining a block is called minting or forging a block. Contrary to PoW miners, PoS validators are not in direct competition against each other in terms of computational power for minting the next block. Instead, the network elects which validator should mint the next block, thus preventing the wasteful process of PoW mining. The selection of the next block's validator is not random. To become a validator, a node has to deposit something as a security deposit, known as its stake. This stake remains reserved and is returned to its owner, along with all the transaction fees of the minted block, only after some time has passed. In case the validator has approved a fraudulent transaction, it loses all transaction fees and its originally pledged stake. This creates a profit-based motivation to follow the protocol. The rationale behind electing a validator proportionally to its stake is straightforward: The more you risk losing, the less likely you are to cheat. This, however, encourages a rich-gets-richer pattern, where the richest nodes are repetitively appointed validators, collecting all transaction fees, and getting even richer. To alleviate this problem, the use of coin age has been devised: A coin's power in increasing a node's chances to be appointed a validator is dependent on the time since the coin was last used as stake. That is, coins that were recently used as stake are of no use in being used again, until a certain time period has passed. Although PoS significantly lowers its ecological impact as a consensus algorithm compared to PoW, it is not without its drawbacks. The operational cost of a PoS miner is negligible, thus enabling a single miner to participate in multiple PoS blockchains using the same machine. As a result, a new attack vector called Nothing at Stake is feasible. In PoW systems, when a blockchain fork occurs, miners are indirectly forced to choose a fork they will continue supporting as they will not be able to participate in multiple PoW blockchains due to the computational capacity required. In PoS systems, this protection is nullified and, as such, nothing prevents a validator from supporting any number or even all the forks of a PoS blockchain simultaneously, as his profits will increase regardless of which fork becomes dominant. Additionally, a tragedy of commons also facilitates the above modus operandi. If the stake of a miner who wishes to initiate a fork in a PoS system is sufficiently large, other miners will join his fork in fear of losing their stake and being unable to overcome and prevent the fork. As this problem exists inherently in the absence of a restrictive resource for operating nodes in a PoS scheme, its solution remains the subject of academic research initiatives. One such initiative is called Ouroboros, led by the Universities of Edinburgh, Connecticut, and Aarhus, and a private organisation named IOHK. They explore a solution based on the forking mechanism of a blockchain. There are a number of systems using various flavours of PoS, including Cardano, Lisk, BlackCoin, Peercoin, and Nxt. Ethereum has announced the intention to switch to PoS.

2.5.1.1.3 Proof-of-Authority (PoA)

Proof-of-Authority (PoA) delegates transaction validation and block creation to a certain set of authorized nodes who are acting as the administrators of the system. This paradigm best fits private blockchains, as it exhibits a centralized operation: denoting who are eligible as administrators (and who are not). Transaction throughput can be high, and blockchain parameters can be fine-tuned to the specific needs of the private networks they serve. However, trust does not emerge from the inherent dynamics of stakeholders, but is rather “outsourced” to the sysadmins guaranteeing the secure and flawless operation of a big enough fraction of the authorized nodes. Most of the PoA-based consensus mechanisms are based on the so-called Byzantine consensus protocols. These protocols stem from the seminal work of Lamport, Shostak, and Pease in the early 1980s and later works of Castro and Liskov. While many Byzantine protocols are being employed elsewhere, e.g., in many cloud databases, their use in DLTs is mainly differentiated through the DLTs being append-only databases.

2.5.1.1.4 Proof-of-Elapsed Time (PoET)

Proof or Elapsed Time (PoET) is a consensus mechanism introduced by Intel, making use of their CPUs’ Software Guard Extensions (SGX) feature, enabling processors to run trusted code that cannot be tampered with. The logic behind PoET is simple. Each participating node waits for a random period, and the first node to finish waiting becomes the validator for the next block. Essentially this is a basic leader election protocol, which uses negligible CPU power to execute. Clearly, its correctness depends on nodes being honest with respect to waiting for a really randomly chosen time period. However, this is guaranteed through code that is trusted based on Intel’s SGX feature. One example of use of PoET is in Hyperledger Sawtooth3, developed primarily by Intel.

2.5.1.1.5 Hybrid approaches

In addition to mechanisms that are pure PoW, PoS, PoA, or PoET, we are starting to see more and more mechanisms that mix some features of these together, e.g., into systems that require both staking a pledge and performing some work. However, as these mechanisms are still being developed and, as far as we can see, none of them are considered well established by the larger community, they mostly fall beyond the scope of this document. However, some examples of such hybrid mechanisms can be found in, e.g., some of the so-called ledger-ofledger approaches, which often seem to combine PoW and PoS mechanisms.

2.5.1.2 Smart Contracts

Smart Contracts bring another groundbreaking innovation to the DLT world. Rather than using DLTs’ decentralized trust model for offering just an immutable decentralized append-only data store, they exploit the mechanisms to provide a tamper-proof decentralized “world computer”. Through smart contracts, DLTs are promoted from special-purpose tools serving a single application (e.g., Bitcoin) to general-purpose platforms, allowing developers to deploy and execute custom code that may implement arbitrary application logic. A smart contract is essentially a program running on a ledger, maintaining some internal state, and updating this state through transactions. To allow transactions to be made, a smart contract exports an API consisting of one or more functions. Read-only functions, i.e., functions that only read the smart contract's program state without changing it, are executed locally on the node that calls them. Calling such functions does not create new transactions to the underlying ledger. Read-write functions, i.e., functions that update the smart contract's state, are executed on all nodes running the ledger, and require new transactions to be entered into the ledger. More specifically, all nodes that validate a block now, or any time into the future, have to execute all smart contract function calls included in that block, to check whether the resulting overall state matches with the state recorded into the block's hash. Even more specifically, smart contracts gain their power through the following fundamental features:

- **State storage**: It is possible for a smart contract to store its own state into a ledger. This makes it possible for a smart contract to store arbitrary data, not just data specific to Bitcoin or some another application.
- **Turing-complete computation**: Many smart contracts enjoy a Turing-complete computation environment. This allows them to implement arbitrary logic for any type of custom ledger applications.
- **Interaction with other contracts**: A smart contract can call functions of other contracts to interact with them.
- **Input from the external world**: A smart contract can get input from external sources, such as what is the current cryptocurrency value in fiat money (euro, dollar, etc.). This can be done indirectly through interaction with other contracts, which, in turn, get updated by (trusted) external organizations pushing frequent updates to them (an "oracle").
- **Input from the ledger itself**: A smart contract is generally allowed to read and use any value in the ledger, such as last block's hash value. For example, the block hash value can serve as a deterministic pseudo-random number generator.

A platform supporting smart contracts essentially allows arbitrary applications to be developed, and to benefit from the same level of trust and transaction irreversibility enforced by a larger community of participating nodes. The possibilities opening up with smart contracts are unlimited.

Smart Contract are considered second generation because they've taken this idea of recording transactions on a blockchain and expanded it to incorporate programming languages.

These second generation blockchains have unleashed smart contracts. These allow for customizable transactions. You can custom create transactions to execute however best fits the needs of the parties involved.

A smart contract is a software that runs on each of the nodes of a blockchain network, so that, due to the characteristics of blockchain, the contract is verified within a distributed trust model, without the need a third party. It is as if we could distribute a contract between a large number of people and all of them had a mechanism to verify that contract and determine the fulfillment of the necessary conditions for the repercussions that are made explicit in it.

The contract is "executed" without the need for a trusted authority or third party, in a new relationship scenario that is possible thanks to the blockchain model. Looking at it in an even more tangible way, a smart contract is a programming code. By means of a programming language, which is not affected by the ambiguities of natural language, the parties can define the object of the contract, the actions that can be performed on it and the application clauses.

The reliable execution of smart contracts in a distributed environment has the potential to define automated processes with contracts that can be enforced on their own and will allow the creation of collaborative ecosystems with extraordinary possibilities. In addition, due to the nature of blockchain, a smart contract is stored in the chain of blocks and distributed among all the nodes of the network, without being able to be altered by any of them.

The reliable execution of smart contracts in a distributed environment has the potential to define automated processes with contracts that can be enforced on their own and will allow the creation of collaborative ecosystems with extraordinary possibilities. In addition, due to the nature of blockchain, a smart contract is stored in the chain of blocks and distributed among all the nodes of the network, without being able to be altered by any of them.

❖ *Characteristics of Smart Contract applications*

Several documents are loaded into the system that make up the transaction (standard documents such as Bill of Lading are a standard part of the software program and will not change);
Therefore, the relevant contract is published by the owner of the asset and the counterparty will negotiate the price / charge directly through the blockchain network.
As each party approves and signs the relevant document in the system, the program approves and moves the transaction to the next stage.
The contract is executed by a computer network using consensus protocols.
Automatic execution and simultaneous loading of information so that all parties can see.

Advantages of the Smart Contract

- Efficiency
    - As the parties comply with the predefined stipulations, the contract will be executed by itself, which increases the efficiency of the custody of the shipment by limiting the intermediaries involved.
    - Tasks that are normally completed manually can also be automated.
- Transparency
    - The information is available to all with the required access code, which limits the risks in the transactions since the counterparts can evaluate the information along the chain.
    - Since the execution of contracts and other tasks are automatic, there is less risk of human errors.
- Security
    - All information is encrypted and can not be altered by the parties as protection against fraud.
    - Higher accuracy combined with greater security will limit disputes regarding the validity of the transaction.
    - Encryption software and increased visibility of all parties will decrease the level of theft and piracy in the industry.
    - It will also facilitate the distribution of responsibility for such events.
    - Cost savings.
    - Large costs related to documentation, procedural delays and errors will be limited.
    - Costs related to several intermediaries will be eliminated or limited.
- Market
    - Everyone can access the blockchain technology.
    - Lower entry barriers will increase competition in the market
    - The parties can develop direct communication without intermediaries
    - Decentralized brokerage system based on blockchain technology
    - Open market for financiers, owners, shippers and carriers
    - Open information on the capacity, cost and estimated delivery times: easier for the parties to evaluate possible transactions
    - World Economic Forum: global trade will increase by 15% and world GDP will increase by 5%
    -

Disadvantages of the Smart Contract

- Special contractual terms
    - Contracts related to maritime transport can be very unique and specific.
    - The blockchain network will need to recognize specific maritime standards and characteristics.
- Flexibility
    - Certain problems during the transaction are usually solved commercially.
    - This could be potentially difficult in a sealed system approach
    - The parties will also have different contractual terms that must be agreed upon and adopted by the blockchain network
    - Global adoption
    - No current government or jurisdiction has implemented this.
    - It will take some time before being fully implemented.

2.5.1.3 Permission Model

The third aspect, the ledgers' policy on which nodes are allowed to act in which roles, places the ledgers into two broad categories: permissionless and permissioned ones.

2.5.1.3.1 Permissionless (Open Ledgers)

In permissionless or open ledgers any computer that has network access may join the ledger, basically taking up any role. That is, it may opt to participate as a validator (miner) to contribute in building consensus, as a verifier (full node) to read and locally verify blocks, or simply as a user and issue new transactions. This model is the most well known one, used in a number of blockchains, including Bitcoin, Ethereum, Litecoin, Monero, Dash, Kadena, and Dodgecoin.

2.5.1.3.2 Permissioned Ledgers

In permissioned ledgers, a node needs to be authenticated and authorized to take up certain roles. For instance, a ledger could restrict the validators to a predefined set of authorized nodes but let any node to locally verify the correctness of the ledger. Other ledgers could also require authentication and authorization to read the ledger, in which case they essentially become private ledgers.

There is a considerable variation in the permission model among ledgers. Furthermore, there is an interplay between the permission model and the consensus protocols: the PoA consensus model is only possible in permissioned ledgers.

2.5.2 Noteworthy Blockchain Implementations

The number of reported cryptocurrencies has exceeded the whopping number of 1600, as of June 2018. Clearly, a large fraction of them are identical to each other from a technical point of view. Hence, there would be no point in even listing them all. Instead of that, we collected a number of well known (but different) ledgers, i.e., ones that may be considered noteworthy, also for their technical implementation. In this section, we present a number of the most notable blockchain implementations, and elaborate on their operation.

2.5.2.1 Public or open DLTs

2.5.2.1.1 Bitcoin

Bitcoin, introduced by pseudonymous Satoshi Nakamoto, undoubtedly earns the first place in this list, being the seminal blockchain design and implementation, as well as the first succeeding fully decentralized digital currency. Bitcoin's blockchain is used exclusively for transactions in its own cryptocurrency (called bitcoin). That is, there is no support for smart contracts or arbitrary applications. Bitcoin does support some elementary scripting; however, the sole purpose of the scripting is limited to providing flexibility in setting the conditions for spending assets, e.g., requiring either a single user's key, or n users' keys, or k-out-of-n users' keys, etc., to dispose of assets. Bitcoin is an open-source protocol running on its permissionless blockchain. It employs a Proofof-Work consensus mechanism, based on double SHA-256 reverse hashing. As an explicit design decision, Bitcoin generates one block per 10 minutes, on average. This is achieved through a difficulty parameter for the Proof-of-Work algorithm, which is automatically adjusted every 2600 blocks to counterbalance global hashrate increase (or decrease), by considering how much faster (or slower) than 10 minutes the average generation of the last 2600 blocks (about 18 days) was. The incentivization mechanism to attract miners consists in allowing them to pay themselves a predefined amount of newly generated bitcoins for each block they mine. This is also the only way bitcoins are being generated. There are significant scalability concerns regarding Bitcoin, as its transaction processing rate of circa 5–7 transactions per second is deemed too low for serving trade at a global scale.

2.5.2.1.2 Ethereum

Ethereum4 was the first — and is to date the largest — deployed blockchain to support smart contracts. It was proposed by Vitalik Buterin in 2013 (white paper), detailed by Galvin Wood in 2014 (yellow paper), and deployed in July 2015. It introduced the notion of smart contracts. Ethereum incorporates its own currency, called ether. Ether is fundamental for Ethereum's operation in two ways. First, it constitutes the incentive for validators (Ethereum term for miners) to contribute their resources to the Proof-of-Work algorithm. Second, it is used to regulate the use of the blockchain's resources by charging for its use. More specifically, ether is needed to pay for gas, a unit used to measure the computation, storage, and bandwidth cost an operation imposes on the blockchain. To invoke a smart contract function, the caller has to specify how much ether he is paying per gas spent, as well as an upper limit on the gas that can be spent. This way, Ethereum can support a Turing-complete virtual machine, called the Ethereum Virtual Machine (EVM), without fearing a denial-of-service abuse of the system: any long- or eternal-running function costs money, and will eventually be killed for having run out of gas. In Ethereum, it has been a design decision to produce a new block every 14–15 seconds on average, with clear user experience benefits stemming from much faster transactions than in Bitcoin. At this block generation rate, there is a non-negligible probability for more than one blocks to be mined in parallel, effectively creating a fork until subsequent blocks determine the winning block and the orphan block(s) among the competing blocks. Additionally, as 14–15 seconds are comparable to the time it takes to disseminate a block to the entire network, receiving a new block a few seconds earlier gives a significant advantage, which would normally favor large pools of nodes that could give higher priority to disseminating new blocks among themselves before spreading them to the rest of the community. To weaken this effect, Ethereum does not only reward blocks that do get accepted into the main chain, but also (valid) orphan/stale blocks that were on the abandoned path of a fork. For this reward to be handed out, a newer block (called nephew) should link to a past stale block (called uncle) in addition to its direct parent. Then, the uncle block receives 87.5% of a new block reward, and the nephew receives the remaining 12.5% of that reward, as an incentive to include it.

2.5.2.1.3 Cardano

Cardano is being developed in two layers that separate the ledger of account values from the reason why values are moved from one account to the other. The Cardano Settlement Layer (CSL) acts as the balance ledger. It uses a Proof-of-Stake consensus algorithm to generate new blocks and confirm transactions. Moreover, CSL supports sidechains for moving assets from the CSL to the Cardano Computation Layer (CCL) and any other blockchains that support the Cardano KMZ protocol, which is used for efficient Simplified Payment Verification (SPV) proofs. CCL (Cardano Computation Layer) is the second layer of the Cardano platform. It contains the information on why transactions occur. Because the computation layer is detached from the CSL, different users of the CCL can create different rules when evaluating transactions. The Cardano team is creating a new programming language to use to develop smart contracts on the CCL, which is called Plutus. The CCL also supports Solidity — used in Ethereum smart contracts — for low-assurance applications. The Cardano ledger will also implement its own proprietary virtual machine, called IELE6 . The VM is based on the LLVM7, along with its own unique low-level language. The core difference between IELE and the Ethereum VM is that IELE is a register-based machine with an unbounded number of registers, while EVM is a stack-based machine with a stack limit of 1024. As direct result of IELE's design paradigm, it supports unbounded integers, enabling easier development of secure smart contracts. The rationale behind the development of IELE is the creation of a uniform low-level language that elegantly translates to and from high-level languages, due to its register-based nature.

2.5.2.1.4 IOTA

IOTA is a rather different type of a distributed ledger, in the sense that it is not based on a blockchain structure. Instead of letting miners confirm transactions in blocks, IOTA decentralizes this process even further, requiring users themselves to confirm each other's transactions. More specifically, each new IOTA transaction has to include links to at least two past transactions, which it approves directly. By linking to them, it also approves indirectly all past transactions approved (directly or indirectly) by them, all the way to IOTA's genesis transaction. This creates a directed acyclic graph (DAG) linking newer transactions to older ones. Contrary to other distributed ledgers, this process does not involve any transaction fees, as the approval of the transactions is made by the users themselves. Attempting to approve a transaction that is either invalid in itself, or which has approved (directly or indirectly) some invalid transaction, will result in getting the transaction eventually being neglected, and therefore dropped. This constitutes a strong motivation for performing a thorough check when approving past transactions, essentially delegating the task of banning invalid transactions and maintaining consensus to the users themselves. The more new transactions approve, directly or indirectly, a given transaction T, the higher is the confidence on T being valid. There is no fixed confidence level for considering a transaction definite; it is subject to the risk level each user deems acceptable. Currently the users are not allowed to pick arbitrary transactions to approve. Instead, a centralized third party, known as the "coordinator", allocates approval tasks to users submitting new transactions. This is considered a temporary measure that is planned to be dropped later on. A core advantage of IOTA concerns its scalability. As registering a transaction requires checking and approving two other transactions, the IOTA system enjoys an inherent elasticity: The higher the transaction rate, the higher the collective capacity of “approvers”. An important issue remains to find the right motivation for ensuring that all pending transactions will be picked for approval, without any of them being left waiting indefinitely. Another unique feature of IOTA is the use of “Winternitz One-Time Signatures”, which are safe even for post-quantum usage. On the other hand, the use of this type of signature results in big transactions, in the order of 10KB, far larger than Bitcoin’s transactions that have the average size 600 bytes. IOTA is still in a beta phase and has received some negative criticism; for example, MIT’s Media Lab criticizes it for implementing its custom cryptographic solutions .

2.5.2.2 Private/Permissioned DLTs

2.5.2.2.1 Hyperledger Fabric

Hyperledger Fabric is one of the first completely permissioned distributed ledger implementations that are designed for enterprise usage. Following the norm of the latest released ledgers, it is a smart contract capable ledger in which a contract is referred to as a chaincode. The core feature of Hyperledger Fabric is the modularity its internal operational structure provides. The various services that contribute to its operations are split into secluded containers that are independent of each other, enabling users to hot-swap integral components of the system, such as the consensus mechanism or the virtual environment enabling the execution of chaincode, with new and completely different ones. This novel modularity enables Fabric to facilitate the operation of chaincode within conventional operating systems. This, in turn, allows developers to create chaincode in general-purpose programming languages, such as Python and Java, with Go being the language of choice for Fabric. The only prerequisite of a fully functional chaincode is the deterministic execution of its functions, as well as the conformity of its specification to the Chaincode Interface, as defined by Hyperledger Fabric. Being a permissioned ledger, Fabric employs a layer of membership delegation through the usage of a Public Key Infrastructure. Each entity within the Hyperledger Fabric network is uniquely identified by its cryptographic key-pair, for which a valid certificate is assigned by the Certificate Authority (CA) operating in the network. Although the CAs are part of the Fabric network, the ledger operators are split into two subcategories, the Orderers and the Peers. The Orderers are in charge of, as their name implies, ordering the transactions within the blockchain network, resulting in an atomic broadcast of new blocks. Orderers implement the consensus mechanism of the ledger and ensure that the ledger is kept in synchrony among all the Peers. The Peers, on the other hand, simply act as hosts of the ledger and are in charge of maintaining the ledger network by keeping a copy of the ledger and allowing clients to interact with it. The Hyperledger Fabric network does not contain any form of a cryptocurrency. Instead, all transactions are treated as chaincode invocations within the network. Whether a chaincode invocation is valid and should be included in a new block is defined by the chaincode itself once it is deployed on the network. Specifically, a chaincode may specify a set of signatures that need to be acquired by Peers operating within the network in order for a new transaction to be submitted on the blockchain that will alter the chaincode’s state. The Peers that are able to endorse chaincode transactions are called Endorser Peers and they provide their endorsement upon successful execution of the chaincode within their respective environment. The definition of a network in Hyperledger Fabric differs from many other ledgers in that it enables multiple ledgers to operate within the same network, via the segmentation of the network into different channels. Each channel retains a separate distributed ledger, which can communicate with the ledgers running on other channels within the same network. This feature enables organisations that operate on the same network to keep private information within their own Peers, by maintaining their own distributed ledger.

2.5.2.2.2 MultiChain

Multichain is a platform for creation and deployment of private blockchains. It supports both intra- and inter-organizational deployment and it offers managed permissions, rapid deployment, unlimited assets, data streams, customizations, and bitcoin compatibility. It is derived from the Bitcoin Core software but it is not limited to the Bitcoin blockchain. MultiChain aims to provide privacy and control in an easy-to-use package so that the deployment of blockchain technology will no longer be a key obstacle. At the same time, advantages of it being a private blockchain are also eminent; no scalability issues, blockchain only contains transactions relevant to the participants. Multichain applies integrated management of user permissions to solve mining, privacy and openness problems. By using private/public key cryptography, which enables any message to be signed by a user and not just transactions. The platform is able to restrict blockchain access to a list of permitted users, embedding a permission protocol in the handshaking process that occurs during the connection of two blockchain nodes. The same principle can also be extended to other operations, such as the right to send and/or receive transactions to a set of addresses. In that way, the MultiChain platform manages to:

- ensure that the blockchain's activity is only visible to selected participants
- introduce controls and permissions on transactions
- enable mining to take place free of Proof-of-Work and the associated costs in a secure way

The risk of a single participant of a private blockchain to monopolize the mining process is mitigated by placing a constraint on the number of blocks which may be created by the same miner within a given window. In addition, as mentioned above, MultiChain is not limited to a specific blockchain, as it is easy to configure (configuration during uptime is reported as future work) and deploy, not just by specialized developers but also by system administrators. Multichain also supports multicurrency, utilizing the same techniques used in CoinSpark, and moves even further by improving on these schemes by integrating support for third party assets directly into the chain's rules.

A main focus of the MultiChain platform is to provide a way for smooth transitions between private blockchains and the bitcoin blockchain in either direction. This is achieved by the following:

- MultiChain is based on a fork of Bitcoin Core, its interface is fully compatible with it, and it can act as a node on the regular bitcoin network.
- It utilizes Bitcoin's protocol, transaction and blockchain architecture, enhancing just the handshaking process.
- It adopts multicurrency and messaging features offered also by the CoinSpark protocol, to enhance bitcoin transactions and to allow applications which use asset tokenization and messaging to move between bitcoin and private blockchains with minimal code changes.

While there does not appear to be any published criticism specifically on Multichain, to us there appears to be at least two problematic areas:

- Any blockchains that use Bitcoin's protocol and block structure are open to 51% attacks by existing Bitcoin miners.
- Using PoW in a private blockchain is not very energy efficient, since in most private blockchains also PoA or PoS can be used.

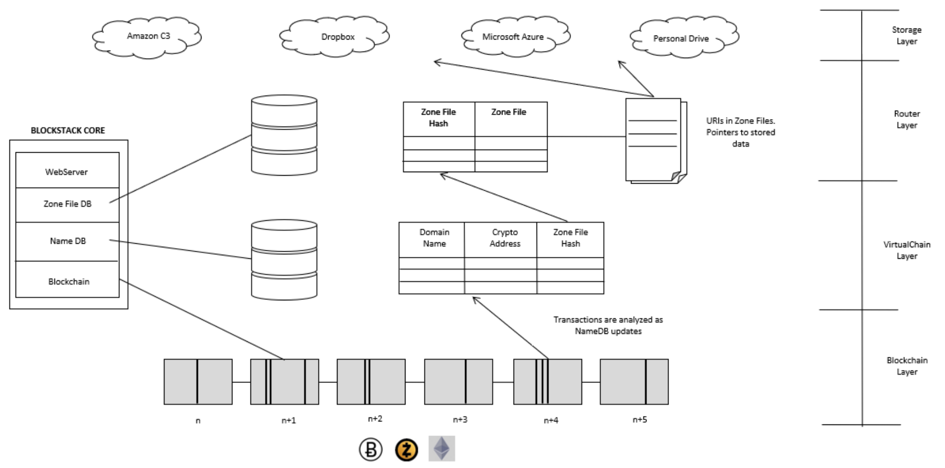
2.5.2.2.3 Corda

It served the purpose of exchanging information between nodes and was fast and easy to adapt for building a customer specific PoC. Corda is a blockchain-inspired distributed ledger by R3, an enterprise software firm owned by banks and other financial institutions. It is meant to serve the purpose of exchanging information between nodes belonging to different companies. It is designed to be fast and easy to adapt for building a customer specific solution. Considering production-ready solution, there appears to be several technological capability gaps in the current open source version:

- To the current stage there is no backwards compatible software available.
- Testing the scalability and performance with 500 devices showed that the overhead to the current storage and network was too high in order to be able to receive millions of devices from different users (geographically distributed). It needs more investigation how architecture and data storage upgrade could handle such volumes together with CORDA Notar.
- There was lack of APIs when using Corda technology.

2.5.2.3 Two DLTs considered for the project

2.5.2.3.1 Blockstack



Blockstack framework

Blockstack [34] proposes to be the new decentralized internet. With Blockstack, users control their data and apps run on their devices. There are no middlemen, no passwords, no massive data silos to breach, and no services tracking us around the internet.

The applications on Blockstack are server-less and decentralized. Developers start by building a single-page application. Then, instead of plugging the frontend into a centralized API, they plug into an API run by the user. Developers install a library called blockstack.js and don't have to worry about running servers, maintaining databases, or building out user management systems.

Personal user APIs ship with the Blockstack app and handle everything from identity and authentication to data storage. Applications can request permissions from users and then gain read and write access to user resources.

Data storage is simple and reliable and uses existing cloud infrastructure. Users connect with their Dropbox, Google Drive, S3, etc.… and data is synced from their local device up to their cloud.

Identity is user-controlled and utilizes the blockchain for secure management of keys, devices and usernames. When a user login with apps, they are anonymous by default and use an app-specific key, but their full identity can be revealed and proven at any time. Keys are for signing and encryption and can be changed as devices need to be added or removed.

Under the hood, Blockstack provides a decentralized domain name system (DNS), decentralized public key distribution system, and registry for apps and user identities.

Blockstack contains the following layers:

Layer 1: Blockchain Layer. Occupies the lowest tier, and serves two purposes: it stores the sequence of Blockstack operations and it provides consensus on the order in which the operations were written. Blockstack operations are encoded in transactions on the underlying blockchain.

Layer 2: Virtualchain Layer. Above the blockchain is a virtualchain, which defines new operations without requiring changes to the underlying blockchain. Only Blockstack nodes are aware of this layer and underlying blockchain nodes are agnostic to it. Blockstack operations are defined in the virtualchain layer and are encoded in valid blockchain transactions as additional metadata. Blockchain nodes do see the raw transactions, but the logic to process Blockstack operations only exists at the virtualchain level.

The rules for accepting or rejecting Blockstack operations are also defined in the virtualchain. Accepted operations are processed by the virtualchain to construct a database that stores information on the global state of the system along with state changes at any given blockchain block. Virtualchains can be used to build a variety of state machines.

Layer 3: Routing Layer. Is separates the task of routing requests (i.e., how to discover data) from the actual storage of data. This avoids the need for the system to adopt any particular storage service from the onset, and instead allows multiple storage providers to coexist.

It uses zone files for storing routing information, which are identical to DNS zone files in their format. The virtualchain binds names to respective hash (zone f ile) and stores these bindings in the control plane, whereas the zone files themselves are stored in the routing layer.

Users do not need to trust the routing layer because the integrity of zone files can be verified by checking the hash (zone file) in the control plane.

In the architecture, nodes form a DHT-based peer network for storing zone files. The DHT only stores zone files if hash (zone file) was previously announced in the blockchain.

A distributed hash table (DHT) is a class of a decentralized distributed system that provides a lookup service similar to a hash table: (key, value) pairs are stored in a DHT, and any participating node can efficiently retrieve the value associated with a given key. Responsibility for maintaining the mapping from keys to values is distributed among the nodes, in such a way that a change in the set of participants causes a minimal amount of disruption. This allows a DHT to scale to extremely large numbers of nodes and to handle continual node arrivals, departures, and failures.

The key aspect relevant of the design is that routes (irrespective of where they are fetched from) can be verified and therefore cannot be tampered with. Further, most production servers maintain a full copy of all zone files since the size of zone files is relatively small (4KB per file). Keeping a full copy of routing data introduces only a marginal storage cost on top of storing the blockchain data.

Layer 4: Storage Layer. The top-most layer is the storage layer, which hosts the actual data values of name-value pairs. All stored data values are signed by the key of the respective owner of a name. By storing data values outside of the blockchain, allows values of arbitrary size and allows for a variety of storage backends.

Users do not need to trust the storage layer because they can verify the integrity of the data values in the control plane.

When we are using a Blockstack client there is control of the data and the ID with a private key. This private key never leaves the device and is meant to stay on the own laptop/phone. As long as no one gets access to the private key, no one can control data or IDs.
Blockstack ensures that all data is signed and verified and (optionally) encrypted end-to-end.

There are two modes of using the storage layer and they differ in how the integrity of data values is verified; the architecture supports both storage modes simultaneously.

- **Mutable Storage** is the default mode of operation for the storage layer. The user's zone file contains a URI record that points to the data, and the data is constructed to include a signature from the user's private key. Writing the data involves signing and replicating the data (but not the zone file), and reading the data involves fetching the zone file and data, verifying that hash (zone file) matches the hash in Blockstack, and verifying the data’s signature with the user’s public key. This allows for writes to be as fast as the signature algorithm and underlying storage system allows, since updating the data does not alter the zone file and thus does not require any blockchain transactions. However, readers and writers must employ a data versioning scheme to avoid consuming stale data.

- **Immutable Storage** is similar to mutable storage, but additionally puts a TXT record in the zone file that contains hash (data). Readers verify data integrity by fetching the data and checking that hash (data) is in the zone file, in addition to verifying the data’s signature and the zone file’s authenticity. This mode is suitable for data values that don’t change often and where it’s important to verify that readers see the latest version of the data value. For immutable storage, updates to data values require a new transaction on the underlying blockchain (since the zone file must be modified to include the new hash), making data updates much slower than mutable storage.

A decentralized high-performance storage system with gaia hub works by hosting data in one or more existing storage systems. These storage systems are typically cloud storage systems, or other backend support as well. Gaia enables applications to access to data it via a uniform API.

The Gaia storage system Will be use to store data on behalf of a user. When the user logs in to an application, the authentication process gives the application the URL of a Gaia hub, which performs writes on behalf of that user. The Gaia hub authenticates writes to a location by requiring a valid authentication token, generated by a private key authorized to write at that location.

2.5.2.3.2 Kadena

Kadena is a viable Proof of Work solution for Scaling Blockchain.

Scalability is one of the most central issues in the current decentralised ecosystem. It receives ample air-time and has a significant share of mind, even the most lay-investor understands on some level that a mature distributed ledger technology (DLT) will be one that has solved for the scaling issue.

Although existing DLT solutions are making headway on this front, an on-chain scaling solution that does not partially or fully compromise the other core tenets of the DLT system is yet to be developed. Kadena introduces Chainweb, a novel architecture that leverages the idea of parallelism braiding multiple PoW chains, to meaningfully increase the network throughput while preserving the core tenets of DLTs. In order to understand the implications of such technology it is important to place it in the context of the current landscape and existing solutions.

❖ *Scaling: A Problem with no simple answers*

Scalability is the ability of distributed ledger technologies to achieve transactional throughput equivalent to its centralised counterparts and meet the ongoing requirements of an expanding network.

Scalability forms one of the three fundamental requirements of a decentralised economy alongside decentralisation and security – the trilemma. The largest hurdle we face in overcoming the scalability bottleneck is achieving the level of required throughput without compromising the other core tenets. Existing technologies have only been able to find a partial balance between the three.

PoW has presented the most viable case for achieving security and trustlessness in the current landscape but its technical implementations concede compromises to scalability, as seen in even industry leaders Bitcoin and Ethereum.

The current issues to scaling PoW Blockchains can be summarized as two deceptively simple issues:

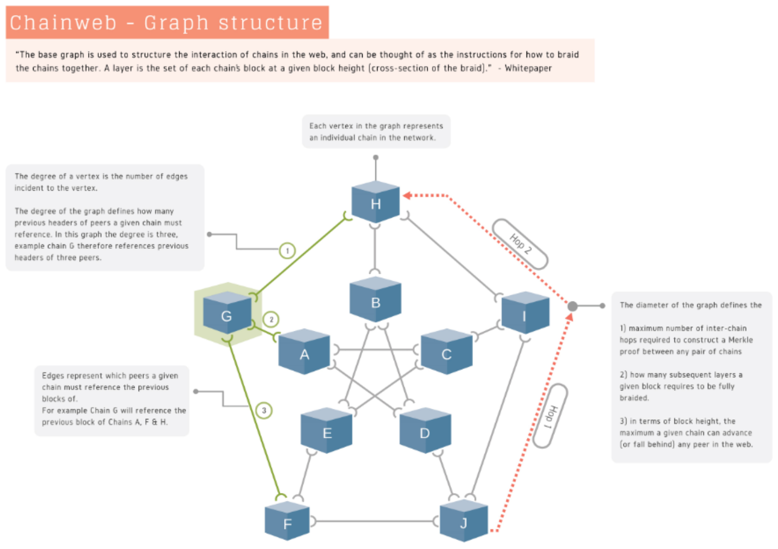
- Time to add a transaction to a block – this refers to the various limitations of a mineable blockchain that stem from the inherent work element of PoW.

- Time taken to reach consensus – time taken for all nodes to verify work done and limitations that arise from the requirement to remain trustless.

❖ *Kadena's Chainweb solution*

Kadena's Chainweb solution, as suggested by the name, is a previously unexplored scaling concept that enables parallel block processing across multiple PoW chains that are braided together into what is a web of chains. Chainweb's value add to the scalability debate is not merely the increase in block processing but rather its ability to do so whilst retaining all aspects of Satoshi Nakamoto's vision of security and trustlessness, altering only the architecture in so far as "braiding" multiple chains together.

PoW has presented the most viable case for achieving security and trustlessness in the current landscape but its technical implementations concede compromises to scalability. This bottleneck is best defined by the trilemma, the inability of any current solution to address all three elements; scalability, security and decentralisation. Chainweb's parallelisation and ability to scale a PoW chain, whilst preserving the other two tenets of the trilemma, achieves a previously untapped set of scalable outcomes.


Chainweb - Graph structure
"The base graph is used to structure the interaction of chains in the web, and can be thought of as the instructions for how to braid the chains together. A layer is the set of each chain's block at a given block height (cross-section of the braid)." - Whitepaper
Each vertex in the graph represents an individual chain in the network.
The degree of a vertex is the number of edges incident to the vertex.
The degree of the graph defines how many previous headers of peers a given chain must reference. In this graph the degree is three, example chain G therefore references previous headers of three peers.
Edges represent which peers a given chain must reference the previous blocks of.
For example Chain G will reference the previous block of Chains A, F & H.
The diameter of the graph defines the
1) maximum number of inter-chain hops required to construct a Merkle proof between any pair of chains
2) how many subsequent layers a given block requires to be fully braided.
3) in terms of block height, the maximum a given chain can advance (or fall behind) any peer in the web.
Hop 2
Hop 1
H
G
B
I
A
C
E
D
F
J
1
2
3

Kadena embeds a full smart contract language (Pact) into its blockchain that can be run as either public (plain text) or private (double-ratchet encrypted) transactions.

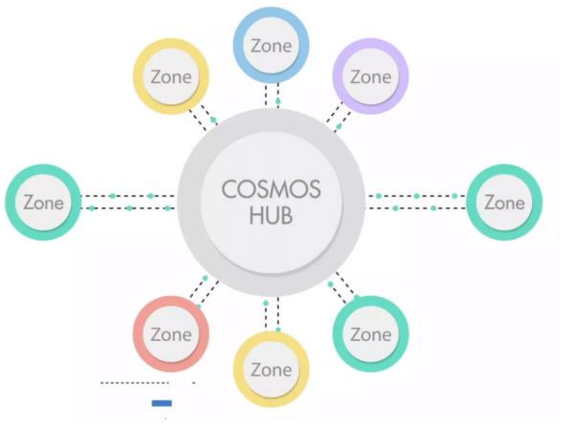
It is a huge step forward in the blockchain space, possibly representing a new-generation of blockchain technology entirely by its introduction of the idea of "pervasive determinism".

Similar to bitcoin, Kadena's blockchain is tightly integrated, and understanding what it is capable of, and what these capabilities imply, requires covering a considerable amount of ground.

2.5.3 Other Blockchains

2.5.3.1 Cosmos

Cosmos aims to become an "Internet of Blockchains" which is going to solve these problems once and for all. Cosmos's architecture consists of several independent blockchains called "Zones" attached to a central blockchain called "Hub".

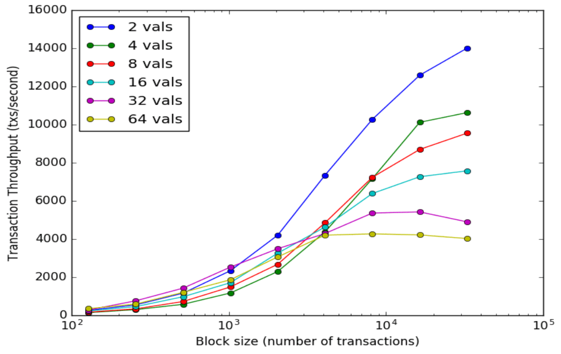


According to the Cosmos whitepaper, *"The zones are powered by Tendermint Core, which provides a high-performance, consistent, secure PBFT-like consensus engine, where strict fork-accountability guarantees hold over the behavior of malicious actors. Tendermint Core's BFT consensus algorithm is well suited for scaling public proof-of-stake blockchains."*

What is Tendermint?
Tendermint is a variant of PBFT i.e. Practical Byzantine Fault Tolerance. A Byzantine Fault Tolerance, or BFT, the system is a system which has successfully answered the Byzantine Generals Problem. We have covered the Byzantine Generals Problem in detail here. To keep things short, for a decentralized peer-to-peer system to function in a trustless manner, it is imperative for them to find the solution to the Byzantine's Generals Problem.

As the cosmos whitepaper states:
"Tendermint provides exceptional performance. In benchmarks of 64 nodes distributed across 7 data centers on 5 continents, on commodity cloud instances, Tendermint consensus can process thousands of transactions per second, with commit latencies on the order of one to two seconds. Notably, the performance of well over a thousand transactions per second is maintained even in harsh adversarial conditions, with validators crashing or broadcasting maliciously crafted votes."

The graph below support the claim made above:



Benefits of Tendermint. Tendermint can handle transaction volume at the rate of 10,000 transactions per second for 250byte transactions.

Better and simple light client security which makes it ideal for mobile and IoT use cases. In contrast, Bitcoin light clients require a lot more work and have lots of demands which makes it impractical for certain use cases.

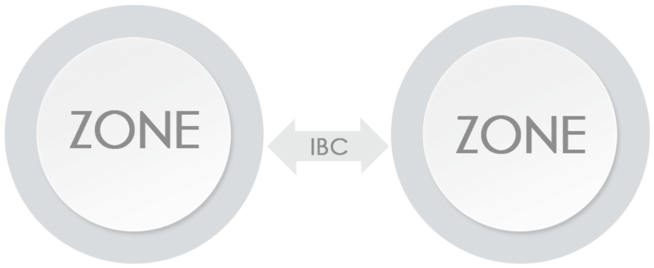
Tendermint has fork-accountability which stops attacks such as long-range-nothing-at-stake double spends and censorship.

Tendermint is implemented via Tendermint core which is an “application-agnostic consensus engine.” It can basically turn any deterministic blackbox application into a distributedly replicated blockchain.

Tendermint Core connects to blockchain applications via the Application Blockchain Interface (ABCI).

Inter-Blockchain Communication. As we have mentioned before, Cosmos’s architecture will follow the Hub and Zones method. There will be multiple parallel blockchains connected to one central Hub blockchain. Think of the Sun and the solar system.

The Cosmos hub is a distributed ledger where individual users or the Zones themselves can hold their tokens. The zones can interact with each other through the Hub using IBC or Inter Blockchain Communication.



This is a very simplified version of how two Zones communicate with each other via IBC.

Cosmos Use Cases. The interoperability achieved by Cosmos has some extremely interesting use-cases:

- **DEX**: Since Cosmos is linking so many blockchains with each other, it goes without saying that it can easily enable different ecosystems to interact with one another. This a perfect setting for a decentralized exchange.
- **Cross chain transactions**: Similarly, one zone can avail the services of another zone through the Cosmos hub.
- **Ethereum Scaling**: This is one of the more use cases. Any EVM based zone which is connected to the Cosmos hub will be, as per the architecture, powered by the Tendermint consensus system as well. This will enable these zones to scale up faster.

2.5.3.2 EOS

EOS are aiming to become a decentralized operating system which can support industrial-scale decentralized applications. The driving force behind EOS is Dan Larimer (the creator of BitShares and Steemit) and Block.One. EOS recently came into the spotlight for their year-long ICO which raised a record-breaking $4 billion.

That sounds pretty amazing but what has really captured the public’s imagination is the following two claims:

- They are claiming to have the ability to conduct millions of transactions per second.
- They are planning to completely remove transaction fees.

Scalability Through DPOS. EOS achieves its scalability via the utilization of the delegated proof-of-stake (DPOS) consensus mechanism, which is a variation of the traditional proof-of-stake. It can theoretically do millions of transactions per second.

So, how is DPOS different from traditional POS? While in POS the entire network will have to take care of the consensus, in DPOS all the EOS holders will elect 21 block producers who will be in charge of taking care of the consensus and general network health. Anyone can participate in the block producer election and they will be given an opportunity to produce blocks proportional to the total votes they receive relative to all other producers.

The DPOS system doesn’t experience a fork because instead of competing to find blocks, the producers will have to co-operate instead. In the event of a fork, the consensus switches automatically to the longest chain.

As you can imagine, the importance of these block producers definitely can't be underestimated. Not only do they take care of consensus, but they take care of overall network health as well. This is why it is extremely important that each and every single vote that has been cast has proper weightage.

This is why, Larimer introduced the idea of Voter Decay, which will reduce the weightage of old votes over time. The only way that one can maintain the strength of votes is by regular voting.

The Voter Decay mechanism leads to two great advantages:

- Firstly, as we have seen time and again, elected officials may become corrupt and change their tune after getting elected. The vote decay system gives the voters a chance to reconsider their vote every week. This keeps the block producers accountable and on their toes.
- Secondly, people simply change over time. Maybe the political beliefs and ideologies that someone has today is completely different than what they had a year ago. The vote decay system will allow people to vote for someone who is more congruent with their newly evolved ideologies.

This has the potential to be a truly revolutionary concept and can change decentralized voting (maybe even voting) forever.

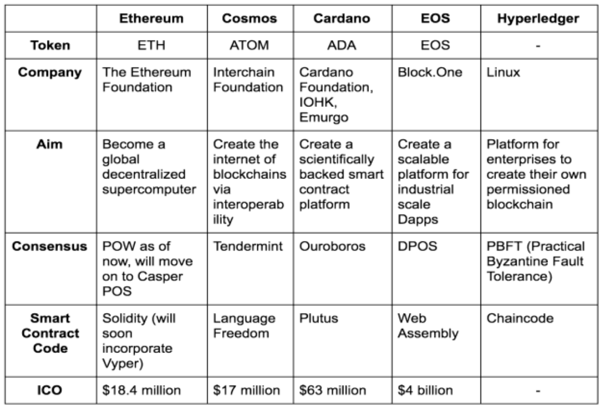
Removal of Transaction Fees. EOS works on an ownership model where users own and are entitled to use resources proportional to their stake, rather than having to pay for every transaction. So, in essence, if you hold N tokens of EOS then you are entitled to N*k transactions. This, in essence, eliminates transaction fees.

On staking EOS tokens you get certain computational resources in exchange. You will get:

- RAM
- Network Bandwidth
- Computational Bandwidth.

EOS tokens, along with payment coins, can also be used as a toll to get all these resources.

2.5.3.3 Comparing some platforms



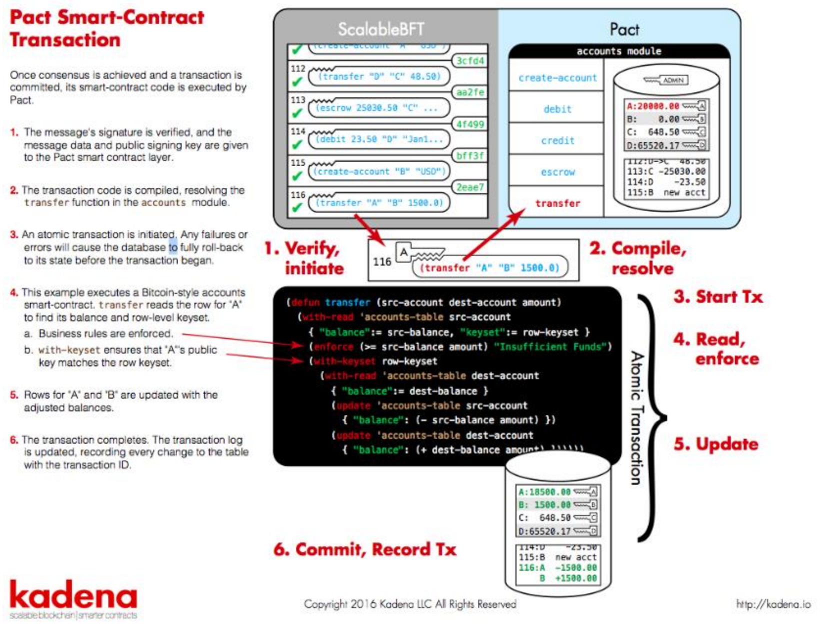
	Ethereum	Cosmos	Cardano	EOS	Hyperledger
Token	ETH	ATOM	ADA	EOS	-
Company	The Ethereum Foundation	Interchain Foundation	Cardano Foundation, IOHK, Emurgo	Block.One	Linux
Aim	Become a global decentralized supercomputer	Create the internet of blockchains via interoperability	Create a scientifically backed smart contract platform	Create a scalable platform for industrial scale Dapps	Platform for enterprises to create their own permissioned blockchain
Consensus	POW as of now, will move on to Casper POS	Tendermint	Ouroboros	DPOS	PBFT (Practical Byzantine Fault Tolerance)
Smart Contract Code	Solidity (will soon incorporate Vyper)	Language Freedom	Plutus	Web Assembly	Chaincode
ICO	$18.4 million	$17 million	$63 million	$4 billion	-

2.6 PACT (Language for Smart Contract)

Pact is a full smart-contract language, the interpreter of which is built in Haskell. In Kadena, every transaction is a smart contract and the Pact smart contract language is open sourced. Pact is database-focused, transactional, Turing-incomplete, single-assignment (variables cannot be changed in their lifetime), and thus highly amenable to static verification.

Pact is also interpreted – the code you write is what executes on-chain – whereas Solidity is compiled, making it difficult to verify code, and also impossible to rectify security issues in old language versions, once compiled. Pact ships with its own interpreter, but can run in any deterministic-input blockchain, and can support different backends, including commercial RDBMS. In the ScalableBFT blockchain, it runs with a fast SQLite storage layer.

In Pact the code is stored in an unmodified form, readable by humans in the general ledger. The declarative, immutable and incomplete Turing design of Pact prevents errors. It contains secure computing functions, as well as encapsulation of database tables in Pact code modules and allows the verification and authorization of signatures.



Once consensus is achieved and a transaction is committed, its smart-contract code is executed by Pact.

1. The message's signature is verified, and the message data and public signing key are given to the Pact smart contract layer.
2. The transaction code is compiled, resolving the `transfer` function in the `accounts` module.
3. An atomic transaction is initiated. Any failures or errors will cause the database to fully roll-back to its state before the transaction began.
4. This example executes a Bitcoin-style accounts smart-contract. `transfer` reads the row for "A" to find its balance and row-level keyset.
   a. Business rules are enforced.
   b. `with-keyset` ensures that "A"'s public key matches the row keyset.
5. Rows for "A" and "B" are updated with the adjusted balances.
6. The transaction completes. The transaction log is updated, recording every change to the table with the transaction ID.

```
(defun transfer (src-account dest-account amount)
  (with-read 'accounts-table src-account
    { "balance":= src-balance, "keyset":= row-keyset }
    (enforce (>= src-balance amount) "Insufficient Funds")
    (with-keyset row-keyset
      (with-read 'accounts-table dest-account
        { "balance":= dest-balance }
        (update 'accounts-table src-account
          { "balance": (- src-balance amount) })
        (update 'accounts-table dest-account
          { "balance": (+ dest-balance amount) }))))))
```

Main features:

- Turing-incomplete, security-oriented design
- Human readable code, in the general ledger
- Atomic execution (transactions)
- Definition and import of modules.
- unique "row-key" + database metaphor
- Expressive syntax and definition of functions.
- "Modular" tables
- Single signature and multiple signature public key authorization
- Multi-step agreements for confidential execution.
- Key rotation support
- Designed for direct integration with industrial databases.

A smart contract in Pact is composed of three central elements: tables, sets of keys and a code module.

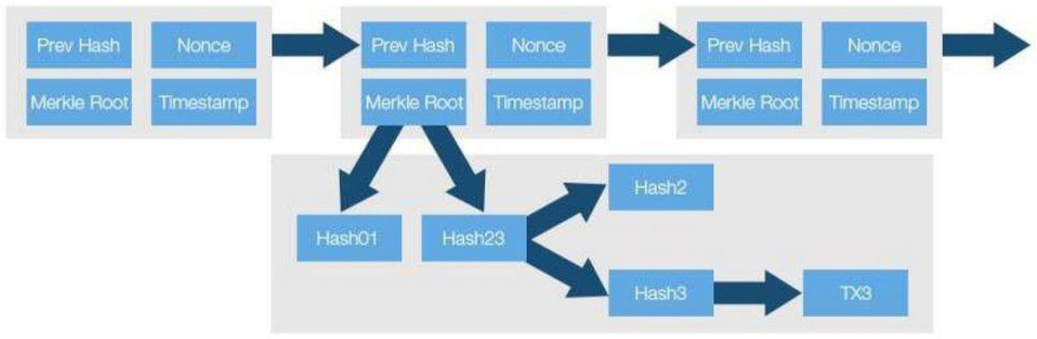
2.7 The future of Blockchain

The concept of blockchain as a standalone technology started gaining popularity in 2015. Prior to that, it was just known as a data structure underlying Bitcoin technology. In Satoshi Nakamoto's white paper, the two words "block" and "chain" appeared together. It was only called "a chain of blocks."

Bitcoin's rise into popularity resulted in it being categorized as Blockchain 1.0 (First Generation). With Ethereum making waves as a decentralized platform for applications that run exactly as programmed, more and more people began to categorize Ethereum as Blockchain 2.0 (Second Generation). Now the market is battling to see who will be named Blockchain 3.0 (Third Generation) .

❖ *What is DAG?*

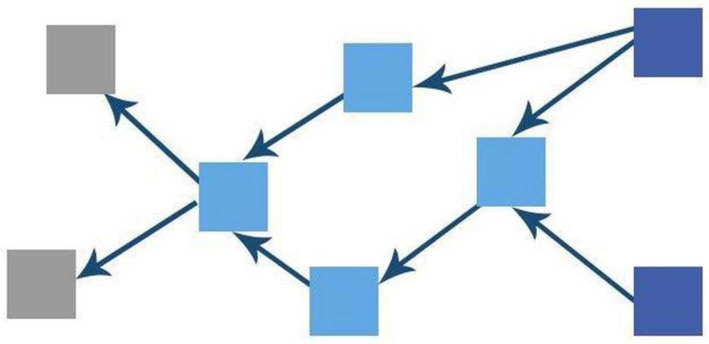
DAG is a directed graph data structure that uses a topological ordering. The sequence can only go from earlier to later. DAG is often applied to problems related to data processing, scheduling, finding the best route in navigation, and data compression.



Bitcoin Blockchain Storage Structure

Bitcoin has always been inefficient due to the proof-of-work (POW) system. Blocks can't be created simultaneously. The linked storage structure allows for only one chain on the whole network. All the transactions occurring around the same time are kept in the same block. Miners then compete for the block validation. One single block is created about every 10 minutes.

The first community to come up with the idea of changing the chain-like storage structure into a DAG of blocks was NXT. If the time of mining remains unchanged, the storage could be extended by X times with X blocks on the network at the same time.



DAG of Blocks Structure

The blockchain combination with DAG still comes from the idea of side-chains. Different types of transactions are running on different chains simultaneously. DAG of blocks still relies on the concept of blocks.

IoT Chain (ITC), IOTA, and Byteball are the blockless projects currently shining in the market. With Bitcoin or Ethereum, the block creation speed is a bottleneck. Bitcoin generates a new block every 10 minutes. Ethereum is better, but it takes around 15-20 seconds for block validation.

But why do we even need a block? On the bitcoin network, many transactions are mined into blocks and the transaction sequence is maintained by the prehashes between blocks. What if you combine blocks and transactions together? Make every transaction directly involved in maintaining the sequences. After the transaction is placed, you can skip the process of mining. This makes it blockless and more efficient.

❖ *Concepts in the DAG Blockchain*

The Double-Spending Issue, from a Probabilistic Perspective. The Bitcoin network uses the UTXO (Unspent Transaction Output) model. Users are only allowed to have one transaction placement under their UTXO. There might be more than one miner who solves the hash function at the same time to acquire the right of block validation. This might develop forks temporarily. The validation of a certain transaction is decided by the number of transactions behind it. The rate of transactions coming back into the network is lower with more transactions behind it, which makes the transaction safer.

The Width of the Network. When each transaction is validated, it needs to be linked to an existing and relatively new transaction on the DAG network. If it links to earlier transactions every time, it would make the network too wide to validate the new transactions. Ideally, the DAG network chooses an existing later transaction to link to when a new transaction happens. The goal is to keep the network width within a certain range that can support quick transaction validation. IOTA also proposed its own algorithm controlling the width on the tangle network.

Quick Transactions. Due to its blockless nature, the transactions run directly into the DAG networks. The whole process is much faster than those of blockchains based on PoW and PoS.

No Mining Involved. There are no miners on DAG networks. The validation of transactions goes directly to the transactions themselves. For users, this means transactions go through almost instantly.

Friendly to Small Payments. With the advancement of DAG, we’re looking at a future where high functioning and minimum transaction fee chains are possible. That means users can send micro-payments without heavy fees like Bitcoin or Ethereum.

One project in China looks to be taking a serious swing at being the leader in this space. IoT Chain (ITC) is built on DAG and can handle over 10,000 transactions per second. It has a strong vision, strong community, and is backed by leading blockchain funds like ChainFunder and FBG. IoT Chain has a solid shot at becoming categorized as Blockchain 3.0.

DAG will be used for applications that require scalability in the thousands of transactions per second. The launch of CryptoKitties clogged the Ethereum network which resulted in slow transactions and high fees. Ethereum has a solution to this called sharding, but it is 5 years out. Applications will soon, I think, be turning to DAG to scale.

2.7 Storage

2.7.1 Cloud

2.7.2 CEPH

Whether you want to provide Ceph Object Storage and/or Ceph Block Device services to Cloud Platforms, deploy a Ceph Filesystem or use Ceph for another purpose, all Ceph Storage Cluster deployments begin with setting up each Ceph Node, your network, and the Ceph Storage Cluster. A Ceph Storage Cluster requires at least one Ceph Monitor, Ceph Manager, and Ceph OSD (Object Storage Daemon). The Ceph Metadata Server is also required when running Ceph Filesystem clients.

Monitors: A Ceph Monitor (ceph-mon) maintains maps of the cluster state, including the monitor map, manager map, the OSD map, and the CRUSH map. These maps are critical cluster state required for Ceph daemons to coordinate with each other. Monitors are also responsible for managing authentication between daemons and clients. At least three monitors are normally required for redundancy and high availability.

Managers: A Ceph Manager daemon (ceph-mgr) is responsible for keeping track of runtime metrics and the current state of the Ceph cluster, including storage utilization, current performance metrics, and system load. The Ceph Manager daemons also host python-based plugins to manage and expose Ceph cluster information, including a web-based Ceph Manager Dashboard and REST API. At least two managers are normally required for high availability.

Ceph OSDs: A Ceph OSD (object storage daemon, ceph-osd) stores data, handles data replication, recovery, rebalancing, and provides some monitoring information to Ceph Monitors and Managers by checking other Ceph OSD Daemons for a heartbeat. At least 3 Ceph OSDs are normally required for redundancy and high availability.

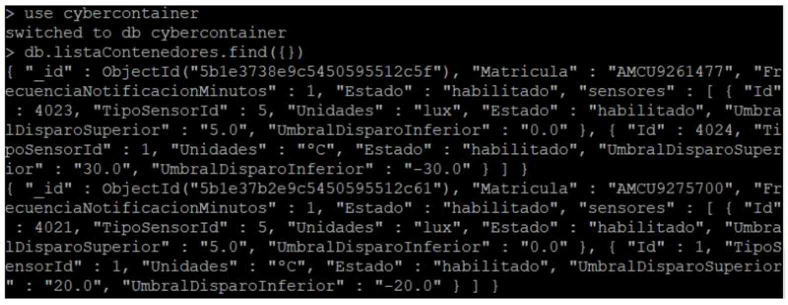
MDSs: A Ceph Metadata Server (MDS, ceph-mds) stores metadata on behalf of the Ceph Filesystem (i.e., Ceph Block Devices and Ceph Object Storage do not use MDS). Ceph Metadata Servers allow POSIX file system users to execute basic commands (like ls, find, etc.) without placing an enormous burden on the Ceph Storage Cluster.

Ceph stores data as objects within logical storage pools. Using the CRUSH algorithm, Ceph calculates which placement group should contain the object, and further calculates which Ceph OSD Daemon should store the placement group. The CRUSH algorithm enables the Ceph Storage Cluster to scale, rebalance, and recover dynamically.

2.7.3 MongoDB

MongoDB is a free and open source database of the NoSQL type, since it is document oriented. MongoDB stores data in the form of dynamic documents with a JSON-like format called BSON (Binary JSON), which allows the data structure to vary over time and between different documents.
A single database can contain different collections, which are the ones that store the documents. It should be noted that each document has its own internal identifier (field "_id"), which is created automatically when the document is created and can not be modified.

MongoDB has its own command console written in Javascript, which allows the user to easily manage their databases through the use of Javascript functions whose use can be found in the official documentation.Asimismo, existen librerías de clientes de MongoDB para diversos lenguajes de programación como Python, Java, C++, C# y Javascript [20].



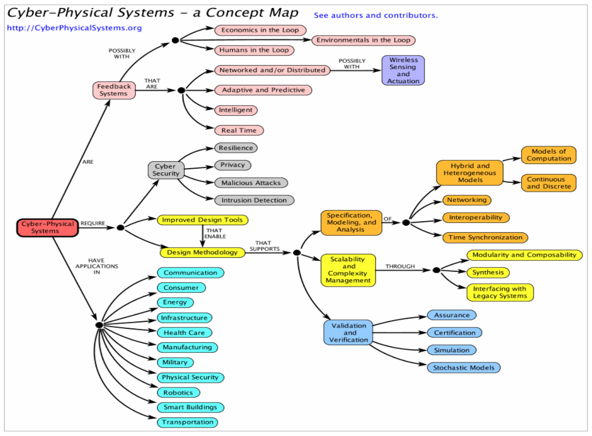
```
> use cybercontainer
switched to db cybercontainer
> db.listaContenedores.find({})
{ "_id" : ObjectId("5b1e3738e9c5450595512c5f"), "Matricula" : "AMCU9261477", "Fr
ecuenciaNotificacionMinutos" : 1, "Estado" : "habilitado", "sensores" : [ { "Id"
 : 4023, "TipoSensorId" : 5, "Unidades" : "lux", "Estado" : "habilitado", "Umbra
lDisparoSuperior" : "5.0", "UmbralDisparoInferior" : "0.0" }, { "Id" : 4024, "Ti
poSensorId" : 1, "Unidades" : "°C", "Estado" : "habilitado", "UmbralDisparoSuper
ior" : "30.0", "UmbralDisparoInferior" : "-30.0" } ] }
{ "_id" : ObjectId("5b1e37b2e9c5450595512c61"), "Matricula" : "AMCU9275700", "Fr
ecuenciaNotificacionMinutos" : 1, "Estado" : "habilitado", "sensores" : [ { "Id"
 : 4021, "TipoSensorId" : 5, "Unidades" : "lux", "Estado" : "habilitado", "Umbra
lDisparoSuperior" : "5.0", "UmbralDisparoInferior" : "0.0" }, { "Id" : 1, "TipoS
ensorId" : 1, "Unidades" : "°C", "Estado" : "habilitado", "UmbralDisparoSuperior
" : "20.0", "UmbralDisparoInferior" : "-20.0" } ] }
```

Ejemplo de una consulta y sus resultados en una consola de MongoDB

2.8 Cyberphysical Systems

The cyber-physical systems (CPS) are obtained from integrating computing, computing and physical processes in the physical objects of the system, to move towards a type of intelligent objects. They are composed of different parties that collaborate together to achieve certain global behavior as if it were a single system. These parts can be communication technologies, embedded systems, sensors, actuators or software systems 21.

In a cyberphysical system, each physical object has its own virtual representation. For this reason, small physical objects are often used, with low consumption and low computing capacity (such as embedded systems), since the computational capacity of the virtual object is used to perform the heavier processes, than in a system normal should perform physical objects.

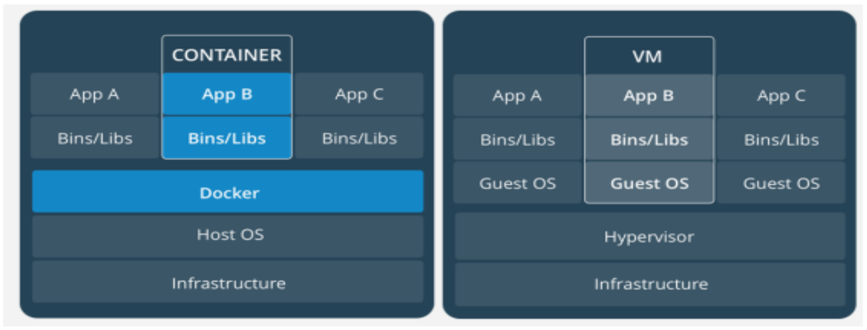


Conceptual map of cyberphysical systems [21]

2.9 Virtualization

Containers are a technology that allows the user to split a machine so that it can run more than one application (in the case of process containers, such as Docker) or instances of the operating system (in the case of machine containers, such as LXD) in the same kernel and hardware of the host machine. In addition, this must be done by maintaining the isolation between these workloads [29].

Virtual machines (VMs) are an abstraction of physical hardware that convert a server into multiple servers. The hypervisor is responsible for allowing multiple virtual machines to run on a single physical machine. Each VM includes a complete copy of an operating system, so they can be expensive to boot [23].



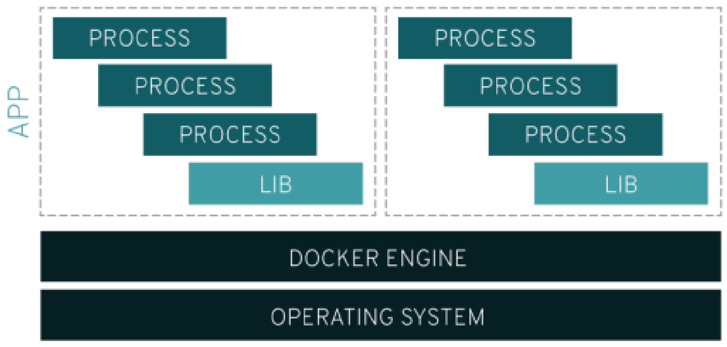
Comparison between the structure of a container (left) and a virtual machine (right) [23]

2.9.1 Docker

Docker is an open source project that allows the user to automate the deployment of applications within process containers. Docker allows a small subset of a file system (only the files needed for an application to run) to be integrated into a template, and then allows this to be implemented, repeatedly, in a lightweight container. The application is packaged so that it is separate from the operating system on which it will be executed. For example, a container built in RHEL can also be run in Ubuntu.

Docker follows the principle of the immutable infrastructure, that is, each container is the same as its template when it is launched, so if a container is restarted all the changes that have been made are lost. In addition, Docker uses a plain text configuration file, called Dockerfile, to describe the elements inside a container, and allows new templates to be produced from existing templates and new files.

Docker has an unusual way of carrying out connections between containers. Each container is not an image of the full-fledged independent operating system, so it does not necessarily have its own IP address. Rather, each container executes one or more processes, each of which will be listening in a port. In the Dockerfile file you can configure which of these ports has to be "exposed", that is, mapped to a host port through a translation. However, in new versions of Docker, each container can be assigned its own IP address and have network interfaces mapped to bridges or overlapping networks. It should be noted that Docker previously used LXC to build its containers, but currently uses libcontainer, which provides similar functionality [22] [23].



Docker containers [22]

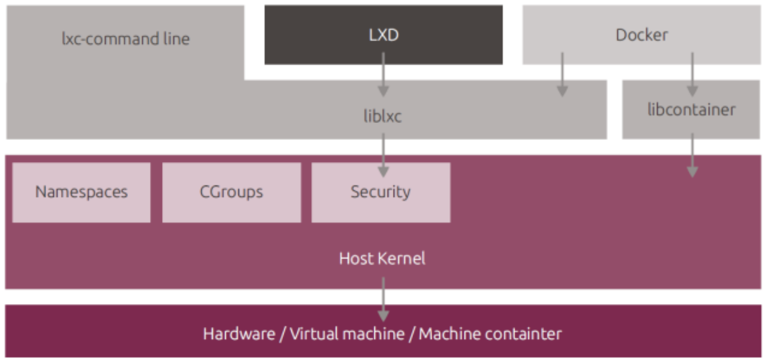
2.9.2 LXD

LXD is an open source tool that provides a means to manage machine containers. It offers the user an experience similar to virtual machines, but using Linux containers instead.

It is necessary to emphasize that LXD is not a rewrite of LXC, but that it is built on LXC to provide a better user experience, and uses LXC through liblxc. In summary, it is an alternative to the tools and the LXC template system with the additional features that come from being controllable through the network.

LXD is made up of three components:

- One Daemon in the whole system, called lxd, which performs all the heavy lifting. It provides a REST API for local use, which can also be exported if the network needs it. In addition, this Daemon communicates with liblxc to create the containers themselves.
- A command line, called lxc, but used to manipulate LXD containers, not LXCs. It communicates with the Daemon through the REST API.
- An OpenStack plugin (Nova LXD) that allows OpenStack to use LXD as a hypervisor, so OpenStack can create instances in LXD in the same way that it would normally create virtual machines that run on a traditional hypervisor such as KVM.



LXD containers in Linux [25]

The main features of LXD are described below:

- Highly scalable
- Secure: by default containers are created without privileges, so any user (even non-root) can throw containers. From there, restriction mechanisms are provided for other users and isolate them from the rest of the system.
- Intuitive: provides a clear and simple API and a clear command line.
- Image-based, with a variety of Linux distributions available. This means that each container has a complete file system, unlike the process containers, which have a composition of templates

2.10 Existing solutions for container tracking systems

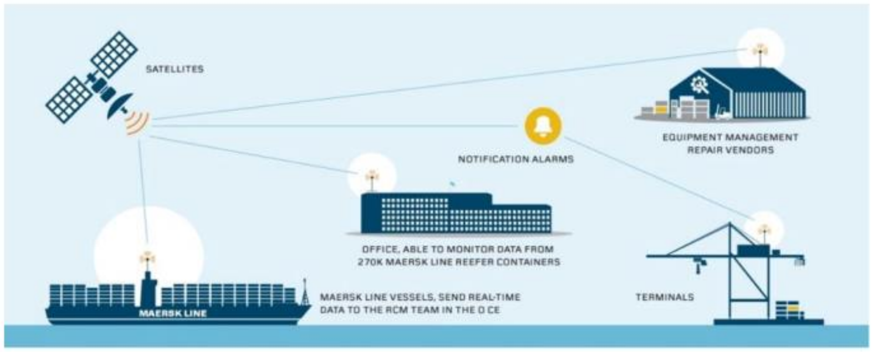
Currently, there are several solutions on the market to monitor in real time aspects of cargo containers, such as their temperature, their position or the integrity of their contents, so some of these solutions are going to be described in this section. It should be noted that all these solutions are proprietary, and therefore only work with elements of the same company that has developed each solution.

2.10.1 Remote Containers Management by Maerks Line

The Remote Containers Management (RCM) solution of the Danish shipping company of containers Maerks Line, allows to monitor the internal conditions of refrigerated containers during the journey of the shipment of the merchandise.

The refrigerated containers include a series of sensors to monitor the condition of the merchandise in real time, as they measure the different parameters that are critical to be able to evaluate the status of refrigerated products: temperature, humidity and the CO2 level of the interior of the container. In addition, threshold values can be predefined for these parameters in order to send notifications if the sensors detect non-optimal values of these parameters, so that the owner of the merchandise can act and avoid the loss of quality of it.

Another important aspect of this solution is that these containers provide information of their position in real time, since they use GPS technology. Therefore, the customer can know at any time the position of their merchandise and exactly when it will arrive at its destination.


SATELLITES
NOTIFICATION ALARMS
EQUIPMENT MANAGEMENT
REPAIR VENDORS
OFFICE, ABLE TO MONITOR DATA FROM
270K MAERSK LINE REEFER CONTAINERS
MAERSK LINE
MAERSK LINE VESSELS, SEND REAL-TIME
DATA TO THE RCM TEAM IN THE O CE
TERMINALS

Operating scheme of the Maersk Line solution [26]

The data collected by the interior sensors of these smart containers are sent in real time by satellite communications. The containers of the same ship transmit this data to a gateway that is in the same vessel, so that finally this gateway will forward the data to a satellite. Next, the satellite will be responsible for relaying this data to the Maerks Line cloud platform. Maerks Line's offices have the capacity to monitor data of the more than 270000 containers of the company that are on a route [19].

2.10.2 Smart Container by Loginno

The Israeli company Loginno has developed an intelligent container with the use of the IoT. This container allows the monitoring of its characteristics in real time: it measures the temperature of its interior, it provides information on its position, it is able to detect the collisions of the container and if its doors have been opened during the journey, which could affect the integrity of the goods it transports. Using this information, this smart container provides notifications to the client whether it detects irregularities or has been intrusions (has been opened without authorization), if it has deviated from its intended route or if there has been a delay with respect to its estimated time of arrival.

Login does not provide a friendly user interface to control all container shipments of the same customer and receive notifications of containers correctly, also provides an app for smartphones for convenience in receiving alerts and notifications. Another important aspect is that it provides an API to be able to use this data on a customer's own platform [27].

2.10.3 Smart Container by Traxens

The French company Traxens has developed a technology to create intelligent containers that allow the owner of the content transported in the container to have a real time visibility of the internal conditions of the container and can receive notifications and alerts of key events when they occur. In addition, it provides a big data platform for the user to have more control over container data.

The solution of intelligent containers of the French company Tranxens is formed by three elements: Traxens-Box, Traxens-Net and Traxens-Hub [28].

- *Traxens-Box*

The Traxens-Box is a monitoring device that is permanently glued to a container and can be used in all types of containers: refrigerated, liquid and dry tanks or standard. The functionality of this device is the collection of GPS position data, temperature, impacts, movement and vibration, and its subsequent transmission to a Traxens-Hub, which is the Traxens data platform. In the same way, it is also possible to connect other types of sensors (humidity, pressure, ...) to a Traxens-Box wirelessly connecting them to the Traxens network, the Traxens-Net.

On the other hand, for the correct operation of a Traxens-Box in a refrigerated container, it is necessary to use an additional unit called Traxens-Link and some cables to connect this unit to the refrigerator controller. In addition, this system makes it possible to send data to the controller of the refrigerator to change the parameters of it, but always under a strict level of security [28].

❖ *Traxens-Net*

The Traxens-Net is Traxens' own network: it is a mesh radio network designed specifically to achieve an optimal ratio between operation and consumption of the devices for the single environment of the containers 28. Its main characteristics are the following:

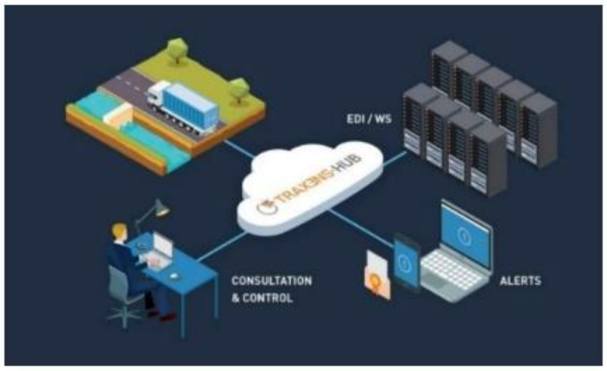
- Distribution of energy between nearby containers through the automatic creation of clusters with the choice of a main container depending on their communication and battery capacities.
- Wide range, even on the largest ships in the world.
- Possibility of adding additional sensors to measure a greater number of parameters, only for one path or permanently.
- Optimal performance in metallic and wet environments.
- Optimal performance in accordance with local radio emission standards.

❖ *Traxens-Hub*

The Traxens-Hub is the Big Data platform that collects all TRAXEBSBOXes data and prepares them to show them to the client in the most useful way for them, facilitating their understanding and making decisions [28].

Its main features are the following:

- Integration with the customer's platform through web services or EDI (electronic data interchange).
- Web-based access.
- Delegation of right of access to the platform, to allow customers or suppliers to participate in the supply chain.
- Send personalized notifications and alerts to users.
- Advanced data analysis tools.



Architecture of the Traxens network [28]

2.10.4 Cellocator's Cellotrack

Cellotrack is a product designed by the Cellocator division of the company Pointer Telocation for advanced asset monitoring and remote monitoring, which consists of a single portable device (includes a long-life battery) that is placed in the object that you want to monitor, as it could be a cargo container. This device is equipped with a long-lasting waterproof housing with an IP67 degree of protection.

This device has an integrated 3D accelerometer for the detection of movement and vibrations, a GNSS receiver that supports GPS and GLONASS to determine the position and the tracking of a route in an exact way, and, in addition, has 2 configurable GPIOs that can be use, among other functions, to connect different types of sensors to measure the different parameters that the user wants to monitor.

Regarding the communications of this device, CelloTrack device uses 3G cellular communications to communicate with the end platform of the user and to send the data collected by the device during the trip to be monitored, so it is necessary to use a sim card of a mobile operator.

Likewise, the device transmits with an adaptive transmission rate, depending on the battery level and the movement of the device.

On the other hand, another product of the same company based on the CelloTrack device is the so-called CelloTrack Container Lock. This product is placed in the lock of the cargo containers and provides information about the container to the user, with the same functionalities as the CelloTrack. In addition, the CelloTrack Container Lock contains a magnetic sensor that detects the opening of the doors of the container, which allows the user to have a strict control of all the times the container has been opened, that is, a greater control over the integrity of the container. content transported by the container [29].

2.10.5 Triton by GlobeArea

Triton is a product designed by GlobeAerea for remote monitoring and monitoring, consisting of a single portable device (including a long-life battery) that is placed in the cargo container. This device is equipped with a waterproof housing of great duration.

It is able to communicate by satellite. It also includes sensors of impact, temperature and light, so you can see the position on a map of all its containers, and manage and receive reports, alerts and customizable events that will provide the necessary information for you to put an effective solution in time, such as: entry and exit of areas of interest, falls, alerts due to excesses or temperature defects, openings in a previously defined area of interest, among many other [30].

2. ARCHITECTURE

❖ *Difficulties in global solutions*

The global expansion of low-power wide area networks (LPWAN) is accelerating its pace, and for good reasons, the demand is there, with a lot of rewards for investments. By creating a market application, we can easily list the different ways in which LPWA technologies can be used to automate processes, track assets, monitor infrastructure and, ultimately, save costs. Its potential is enormous.

❖ *Fragmentation in the ecosystem*

But as we begin to develop a global application, we find ourselves struggling to face a highly fragmented technological, commercial and regulatory landscape. Currently we have different regional priorities for the initial implementation of LPWA technology.

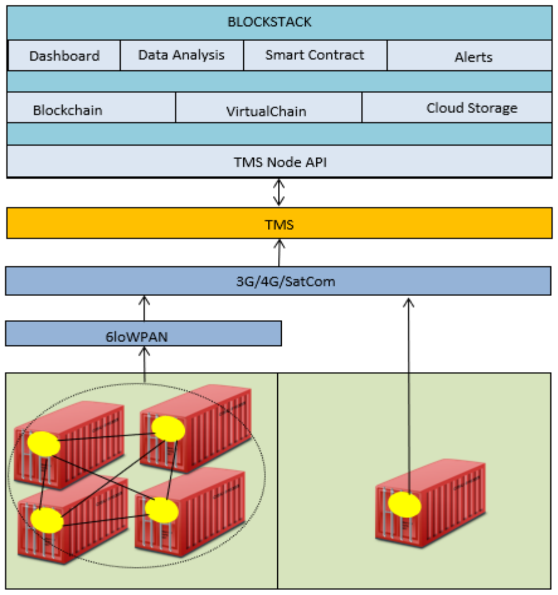
In addition, the way spectrum is allocated within a country or region is highly fragmented into dozens of frequency bands. One way to solve this problem is to manage multiple product variants around the world. While this may address the fragmentation of the network from a regional point of view, it does not reach the enormous efficiencies offered by a global solution that offers connectivity essentially everywhere, all the time and over long periods of life.

❖ *Rationalize and simplify*

The development of successful LPWA solutions for global markets requires a flexible hardware and software solution that can be trusted wherever your devices are deployed. As the global coverage map evolves, it will make it easier to implement an application that works in all four corners of the world.

3.1 General System Architecture

Below you can see the general architecture of the final container management system.



3.2 System Description

3.2.1 Solution LPWAN with star topology with LoRaWan

The first stage is the installation in containers of IoT devices with capacity to measure temperature, relative humidity, concentration of gases (CO2, O2, ethylene ...), vibration, impact, proximity, location (GPS), opening of the enclosure, luminosity, weight, or another. Each of the sensors have thresholds established by the IoT device administration service provided by Paradigma through the web platform.

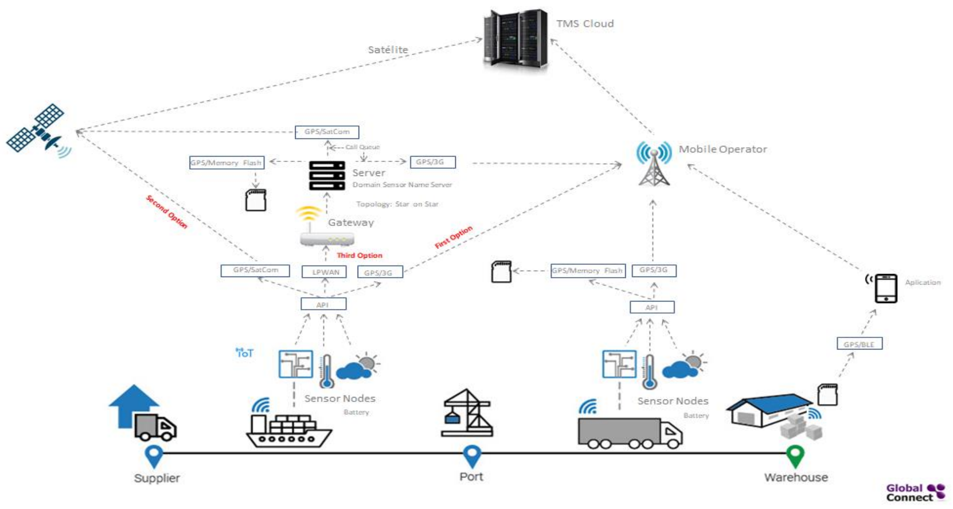
In addition to the minimum and maximum thresholds, you must also configure the frequency of capture/transmission of sensor data.

These IoT devices have a built-in GPS module for localization, and communication modules for 3G/4G mobile operators, Satcom satellite telephony, Bluetooth BLE, and a wireless network module LPWAN. Additionally, a flash memory storage system will be implemented in case of no telephone coverage.

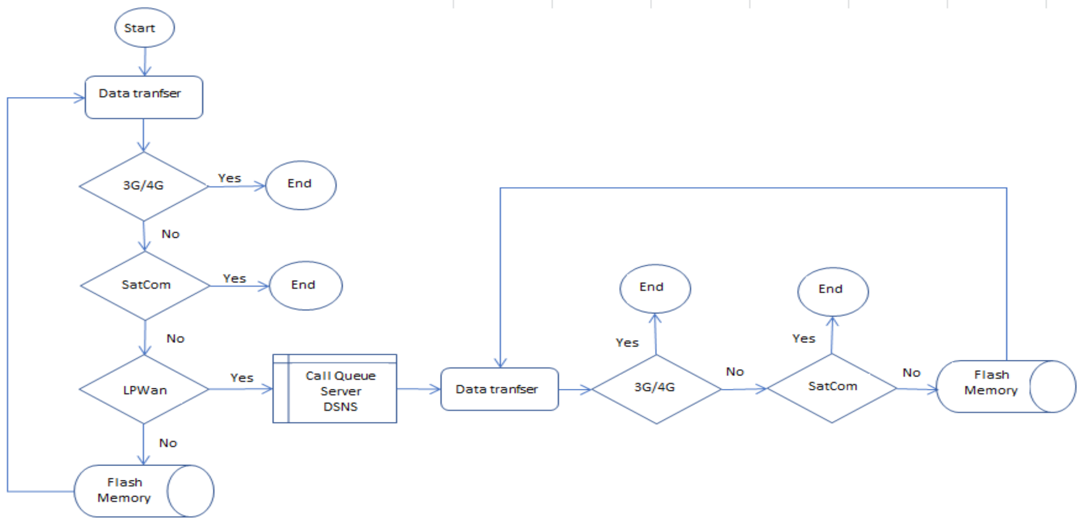
In addition, a star-topology network containing gateways and a DSNS server should be implemented on the ship, the server will also have the GPS module for the location, and communication modules for 3G/4G mobile operators, Satcom satellite telephony, and a flash memory storage system in case of no telephone coverage. The server also has a call queue.

The sensors must be able to communicate both on the high seas and on land, that is, the system is multimodal.

The following figure gives us a vision of the dispotivos and their communication:



The IoT device will communicate in the first instance with the mobile operator to transmit the data, if it fails to do so it will be communicated by means of satellite telephony to transmit the data, if it does not succeed because of being without coverage or being very low on the stack of containers, which prevents communication, will transmit the data to the DSNS server through the LPWAN protocol (wireless network) so that the server connects to mobile or satellite telephony, in case the server can not be connected either, the data will be stored in flash memory waiting for coverage to be transmitted.



3.2.2 Solution 6loWPAN with mesh topology with WI-SUN

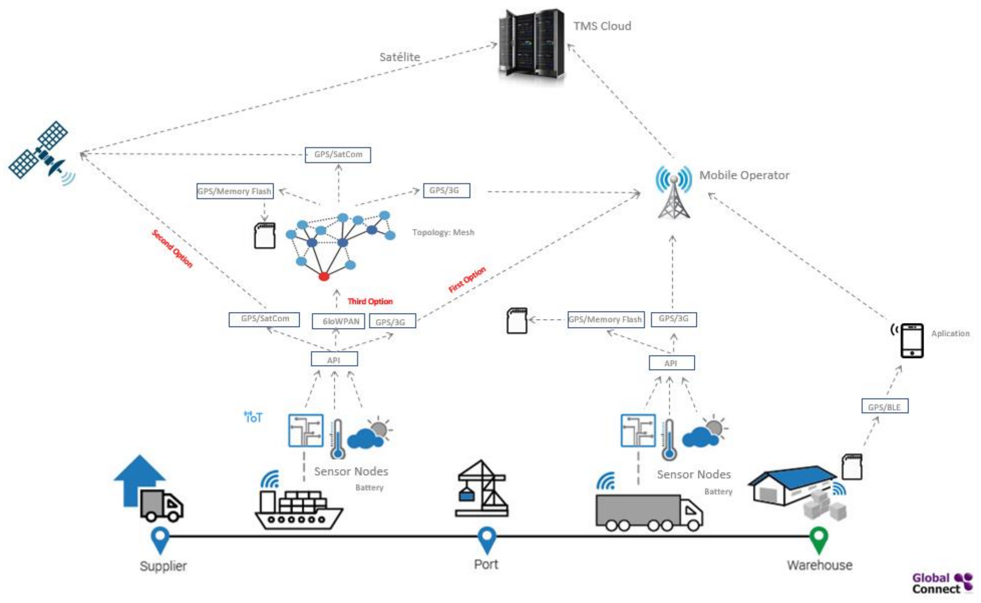
The first stage is the installation in containers of IoT devices with capacity to measure temperature, relative humidity, concentration of gases (CO2, O2, ethylene ...), vibration, impact, proximity, location (GPS), opening of the enclosure, luminosity, weight, or another. Each of the sensors have thresholds established by the IoT device administration service provided by Paradigma through the web platform.

In addition to the minimum and maximum thresholds, you must also configure the frequency of capture/transmission of sensor data.

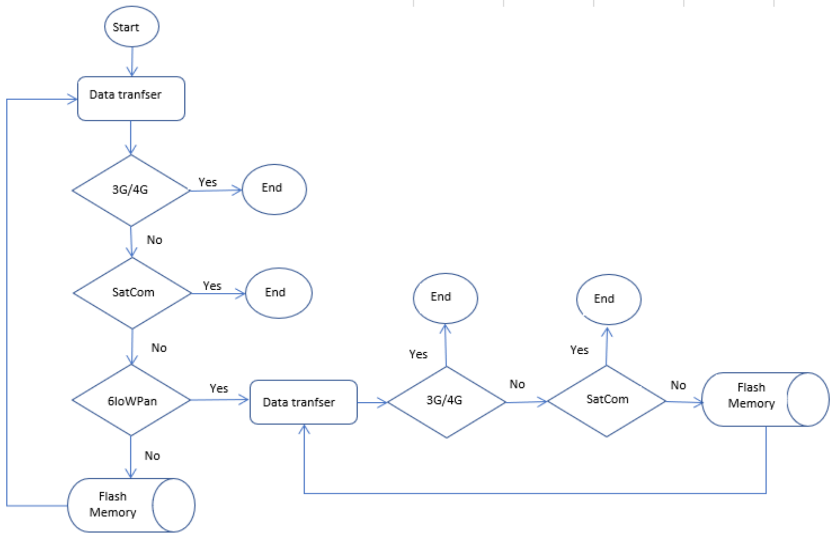
These IoT devices have a built-in GPS module for localization, and communication modules for 3G/4G mobile operators, Satcom satellite telephony, Bluetooth BLE, and a wireless network module 6loWPAN. Additionally, a flash memory storage system will be implemented in case of no telephone coverage.

The sensors must be able to communicate both on the high seas and on land, that is, the system is multimodal.

The following figure gives us a vision of the dispotivos and their communication:

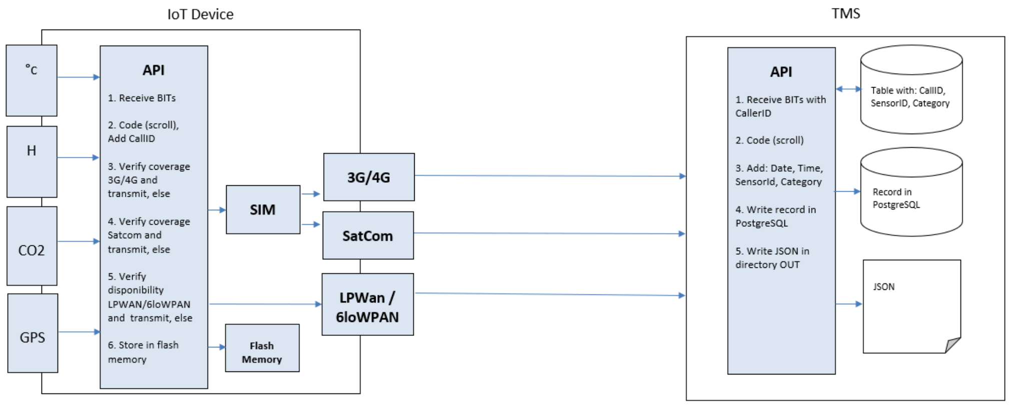


The IoT device will communicate in the first instance with the mobile operator to transmit the data, if it fails to do so it will be communicated by means of satellite telephony to transmit the data, if it does not succeed because of being without coverage or being very low on the stack of containers, which prevents communication, will transmit the data to the nearest device through the 6loWPAN protocol (wireless network), in case no nearby device can be found, the data will be stored in flash memory waiting for coverage to be transmitted.



3.2.3 The IoT devices and the TMS central station

Each device has a CallID, an associated telephone number (SIM card), it must also be considered that the devices can consist of one or more sensors, for this reason the devices must be categorized, since the TMS Server (central station) must understand what IoT device sends data, what is the value of this data and its category, the category will tell us how many measurements (sensors) the IoT device contains, in such a way that the TMS knows how many measurements it should receive from the device, and by means of a formula scroll will be received.


IoT Device
°c
H
CO2
GPS
API
1. Receive BITs
2. Code (scroll), Add CallID
3. Verify coverage 3G/4G and transmit, else
4. Verify coverage Satcom and transmit, else
5. Verify disponibility LPWAN/6loWPAN and transmit, else
6. Store in flash memory
SIM
Flash Memory
3G/4G
SatCom
LPWan / 6loWPAN
TMS
API
1. Receive BITs with CallerID
2. Code (scroll)
3. Add: Date, Time, SensorId, Category
4. Write record in PostgreSQL
5. Write JSON in directory OUT
Table with: CallID, SensorID, Category
Record in PostgreSQL
JSON

We assume that, on the one hand, we have various IoT devices that can broadcast diverse information, that are equipped with phone numbers and can make phone calls.

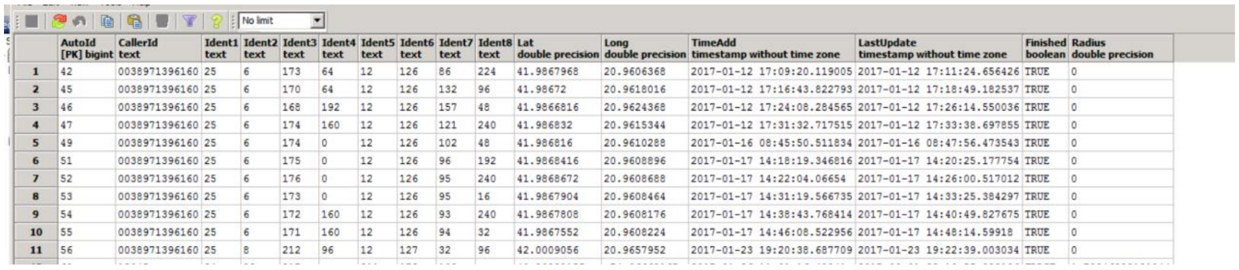
And, on the other hand, we have a centralized station (TMS) with a certain number of telephone numbers that is used to retrieve the information of telephone calls made to this station.

Therefore, on the side of the devices, you must somehow encode the information you want to transmit. This coding must be done through a list of subsequent telephone calls at the central station, which then uses this list of calls to decode the information obtained and sent by a device.

The central station has a database of all the IoT that can make calls, which includes their telephone numbers, along with the category to which they belong. In this way, the phone number (CallID) that calls automatically will give us information about the sensor identification (which sensor) and the category (type of data sent). It means that we do not need coding for these two elements. The only information that needs coding is the value of the data. As there are more calls to code the data, we must identify which call belongs to which part of the information.

The TMS when it recognizes a call, executes a script that has the telephone number and the extension as input arguments.

This script then decodes the value (extension) and places the decoded value in a database (in this case it is implemented in PostgreSql) that contains all the data emitted for different sensors:



	AutoId [PK] bigint	CallerId text	Ident1 text	Ident2 text	Ident3 text	Ident4 text	Ident5 text	Ident6 text	Ident7 text	Ident8 text	Lat double precision	Long double precision	TimeAdd timestamp without time zone	LastUpdate timestamp without time zone	Finished boolean	Radius double precision
1	42	0038971396160	25	6	173	64	12	126	86	224	41.9867968	20.9606368	2017-01-12 17:09:20.119005	2017-01-12 17:11:24.656426	TRUE	0
2	45	0038971396160	25	6	170	64	12	126	132	96	41.98672	20.9618016	2017-01-12 17:16:43.822793	2017-01-12 17:18:49.182537	TRUE	0
3	46	0038971396160	25	6	168	192	12	126	157	48	41.9866816	20.9624368	2017-01-12 17:24:08.284565	2017-01-12 17:26:14.550036	TRUE	0
4	47	0038971396160	25	6	174	160	12	126	121	240	41.986832	20.9615344	2017-01-12 17:31:32.717515	2017-01-12 17:33:38.697855	TRUE	0
5	49	0038971396160	25	6	174	0	12	126	102	48	41.986816	20.9610288	2017-01-16 08:45:50.511834	2017-01-16 08:47:56.473543	TRUE	0
6	51	0038971396160	25	6	175	0	12	126	96	192	41.9868416	20.9608896	2017-01-17 14:18:19.346816	2017-01-17 14:20:25.177754	TRUE	0
7	52	0038971396160	25	6	176	0	12	126	95	240	41.9868672	20.9608688	2017-01-17 14:22:04.06654	2017-01-17 14:26:00.517012	TRUE	0
8	53	0038971396160	25	6	173	0	12	126	95	16	41.9867904	20.9608464	2017-01-17 14:31:19.566735	2017-01-17 14:33:25.384297	TRUE	0
9	54	0038971396160	25	6	172	160	12	126	93	240	41.9867808	20.9608176	2017-01-17 14:38:43.768414	2017-01-17 14:40:49.827675	TRUE	0
10	55	0038971396160	25	6	171	160	12	126	94	32	41.9867552	20.9608224	2017-01-17 14:46:08.522956	2017-01-17 14:48:14.59918	TRUE	0
11	56	0038971396160	25	8	212	96	12	127	32	96	42.0009056	20.9657952	2017-01-23 19:20:38.687709	2017-01-23 19:22:39.003034	TRUE	0

The average time needed to establish call is approximately 5 seconds. Taking that in our case we have available the phone extension ranging from 0000 until 9999, with one call we can transfer approximately 13.3 bits. From this we can conclude that for this communication channel the capacity is 13.3 bits / 5 sec = 2.66 bits/sec.

In this way, with 11 calls we have the data of one container (4 calls for longitude, 4 calls for latitude, 1 call for temperature, 1 call for relative humidity and 1 call for carbon dioxide). Totalizing 55 seconds in the case of mobile telephony, 99 seconds in the case of satellite telephony (placing a delta of 4 seconds with respect to mobile telephony).

For each phone call, a new record is added, filling in one by one all the values for each identifier separately. The device is programmed in such a way that it initially sends the longitude and latitude values first.

For each phone call, a new record is added, filling in one by one all the values for each identifier separately. The device is programmed in such a way that it initially sends the longitude and latitude values first.

For each call, a JSON is generated, which must be saved in the OUT folder:

Schema example

```
{
  "DTEvent": string,
  "EventsContainer": [
     {
        "CallID": string,
        "EventsSensor": [
           {
              "ContainerSensorId": number,
              "ContainerCategorySensorId": number,
              "Value": string,
              "PositionsGPS": {
                 "coordinates": [
                   number,
                   number
                 ]
                 "type": Point
              },
              "DTregistry": string
           }
        ]
     }
   ]
}
```

Event example

```
{
  "data": [
    {
         "DTEvent": "2018-10-21T12:34:00:7992",
         "EventsContainer": [
            {
               "CallID": "9835676546",
               "EventsSensor": [
                  {
                     "ContainerSensorId": 4019,
                     "ContainerCategorySensorId": 34,
                     "Value": "1",
                     "PositionsGPS": {
                        "coordinates": [
                          -0.308595436,
                          39.43556368
                        ]
                        "type": Point
                        },
                        "DTregistry": "2018-10-21T10:30:44:00Z",
                    },
                    {
                        "ContainerSensorId": 4020,
                        "ContainerCategorySensorId": 34,
                        "Value": "14",
                        "PositionsGPS": {
                            "coordinates": [
                              -0.308595436,
                              39.43556368
                            ]
                            "type": Point
                        },
                        "DTregistry": "2018-10-21T10:33:59:00Z",
                    }

                ]
            }
        ]
    }

  ]
}
```

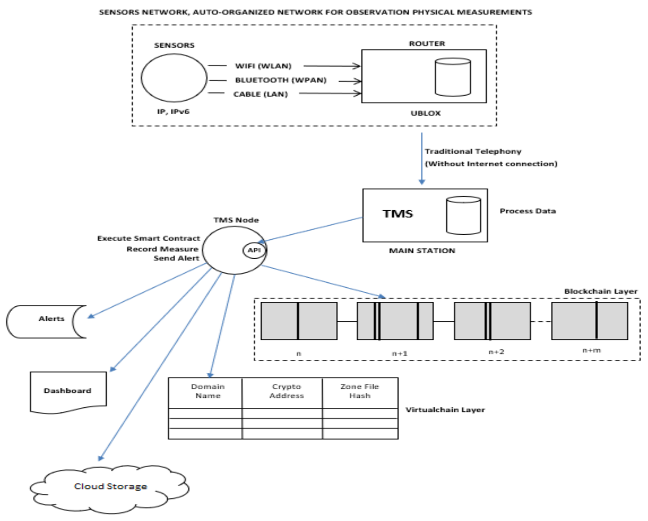
3.2.4 TMS central station and the TMS Node API

The application has the ability to communicate between the TMS server and Blockstack using the HTTP protocol and JSON files (web services).

The TMS server processes and sends the received data from the IoT devices through a specialized Peer to Peer Network based in Atlas, but modified for this solution, renamed as “Globalconnect network”. The components of this network are the TMS Server and several TMS nodes that could be deployed to cover different territories. The TMS server decodes and sends the data of the measurements in XML or JSON format to the *TMS Node* that is in the Atlas Network of the BlockStack of the *TMS Node* ID.

The TMS node receives the data from the TMS Server in XML or JSON format and performs the following actions:

1. Parser the JSON file
2. Relate CallID with ID Container and Bill of lading number
   Note 1*: Each container is identified by a unique identifier. This identifier has a specific format: three uppercase letters followed by a U, to end with seven digits (for example, BMOU8710633).*
   Note 2*: You must have previously related the CallID of the IoT device with the ID of the container and with the Bill of Lading.*
3. With BL number, look for Smart contract associated to BL and retrieve and execute the *Smart Contract* using the received information.
4. Compare the received measurement with the *Smart Contract* rules.
   a. If does NOT comply then:
      - Records the issue in Blockstack (Virtualchain layer and user Storage layer) and in the underlying Blockchain.
      - Send specific user alerts, to be determined (SMS, eMAIL).
   b. If it complies then:
      - Records only in Blockstack (virtualchain layer and user storage layer).



The *TMS Node* API contains classes and methods that allow functionalities such as: search user on which the data is stored, apply the rules of the Smart Contract (associated with the BL), determine if it complies or if it does not comply, write measures in Blockchain and/or Blockstack, send alerts via SMS and/or Email.

The parameters needed are:

- Date and time of shipment (by TMS)
- Date and time received by the *TMS Node*
- Caller ID (from where it is send, to look for user)
- Record type (if temperature, humidity, location, etc ...)
- The actual measurement received.

The CallID allows to know the ID of the container and the folio of the Bill of Lading.

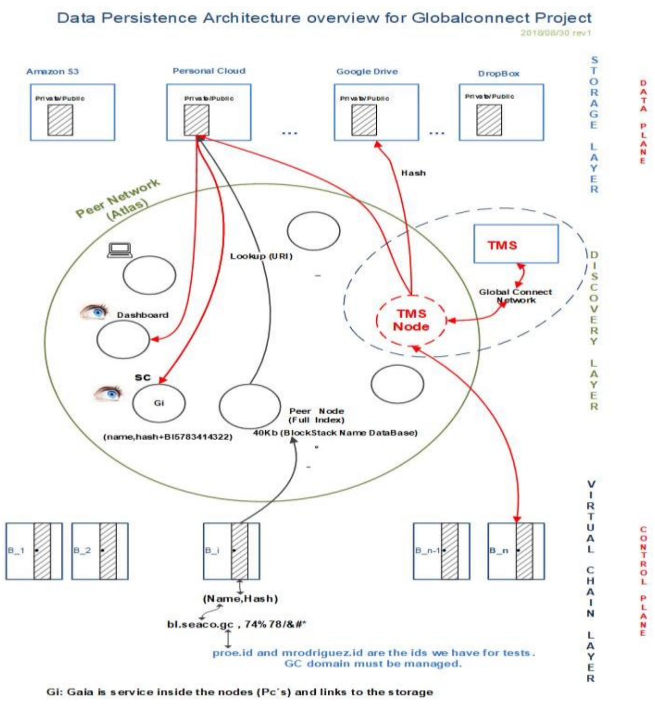
3.2.3 Blockchain with Blockstack

It is a closed ecosystem, a private and permissioned network. The members are invited to join and keep a copy of the ledger Blockchain technology.

The access to the respective network is be restricted and security is thus heightened.

The architecture is based on the Blockstack framework that implements the decentralized authorization through a DNS and PKI, can be used to share IoT device node addresses and data without a central authority in a trustless manner, and distributed storage in the cloud, which allows us to have scalability.

Blockstack domains (called BNS) are not registered on the traditional DNS run by an organized called ICANN. Instead they're registered on a blockchain in a fully decentralized way. This means that Blockstack domains are truly owned by their owners and cannot be taken away. All Blockstack domains have public keys by default (public keys are required to own the domains).



Our implementation has three components:

- A *blockchain*, implemented using *virtualchains*, is used to bind digital property, like domain names, to public keys. Blockstack's blockchain solves the problem of bootstrapping trust in a decentralized way i.e., a new node on the network can independently verify all data bindings.

- A peer network, called *Atlas*, gives a global index for discovery information and

- A decentralized storage system, called *Gaia*, provides high-performance storage backends without introducing central trusted parties.

The architecture decouples the security of name registration and name ownership from the availability of data associated with names by separating the control and data planes.

Introduces four layers, with two layers (*blockchain* layer and *virtualchains* layer) in the control plane, and two layers (routing layer and data storage layer) in the data plane.

The control plane consists of a *blockchain* and a logically separate called a *virtualchains*. In the *blockchain* and in the *virtualchains* (Control Plane) is defines the protocol for registering human-readable names, creating (name,hash) bindings (by example: bl.seaco.gc, x7@#34..........) and creating bindings to owning cryptographic keypairs.

Virtualchains are like virtual machines, where a specific VM like Debian 8.7 can run on top of a specific physical machine. Different types of *virtualchains* can be defined and they run on top of the specific underlying blockchain.

Virtualchains operations are encoded in valid *blockchain* transactions as additional meta-data.

Blockchain nodes do see the raw transactions, but the logic to process virtualchain operations only exists at the virtualchain level.

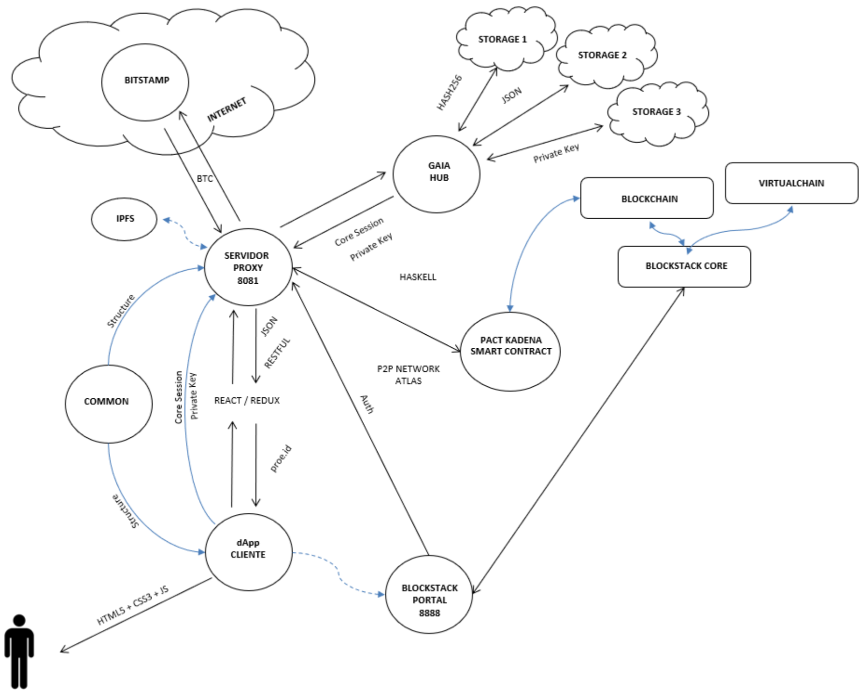
The data plane is responsible for data storage and availability. It consists of:

a) zone files for discovering data by hash or URL, and
b) external storage systems for storing data.

Data values are signed by the public keys of the respective name owners. Clients read data values from the data plane and verify their authenticity by checking that either the data’s hash is in the zone file, or the data includes a signature with the name owner’s public key.

This hides the individual APIs for storage backends and exposes a simple PUT/GET interface to Blockstack users. Looking up data for a name, like werner.id, works as follows:

- Lookup the name in the virtualchain to get the (name, hash) pair.
- Lookup the hash (name) in the Atlas network to get the respective zone file (all peers in the Atlas network have the full replica of all zonefiles).
- Get the storage backend URI from the zonefile and lookup the URI to connect to the storage backend.
- Read the data (decrypt it if needed and if you have the access rights) and verify the respective signature or hash.



3.2.5 Smart Contract with PACT

The traditional Bill of Lading (BL) will be complemented with a *Smart Contract* on the BlockStack and KadenaIO Pact. Each BL could have a completely different *Smart Contract* as the container cargo could be different. This method will provide a high level of security, assurance of compliance of quality measures, traceability, flexibility and a more efficient process to archive past transactions – all key issues in the logistics industry.

The Bill of Lading is the core of the system and is associated with a Blockstack user (for example proe.id) and a *Smart Contract* that evaluates service compliance. The *Smart Contract* gets executed whenever new measurement information is received evaluating if that information is complying with the rules inicially defined.

In addition, the BL is associated with a container and associated with several communication possibilities such as satellite telephone, mobile telephone, among others (PSTN, GSM, etc ...).

Finally, a BL will also be associated with the container enrollment and the IoT devices and sensors.

The system before the association, must verify that the appropiate IoT device is available or occupied.

The user node that assigns the BL with the associations described above (for example proe.id) must top-up BTC in its own application BTC account and in order to operate the system will transfer funds (BTC) to the *Smart Contract* associated with the service. The service requieres funds to operate.

The *Smart Contract* is used to record information. In this way, each time a measure is received, the *Smart Contract* is executed.

From the above it follows that the following information will be stored in the underlying Blockchain:

- Assignment of BL.
- Non-compliance with Smart Contract.
- Termination of the service associated with the BL.

The user node that assigns a IoT device to the BL (for example, proe.id) will have to pay also to the *TMS Node* account a guarantee for returning the of the IoT device, so that it registers that event in the Blockchain.

Once the user who has previously assigned, the service is completed and the IoT device is returned, the *TMS Node* returns the guarantee to the original sender account.

The user who assigns and requests the traceability service authorizes the *TMS Node* to write the storage layer associated with the service. And this same authorizes other users to read the same data.

❖ **Smart Contract Events**

- Born/ Smartcontract starts and price determination
- Link Sensor with container (Subscription)
- Initial Tracking / Location Data
- Voyage Tracking
- Arrived to Destination
- Cargo Accepted/Rejected (proof of delivery)
- Return of IoT Device/ Smartcontract ends

3.2.6 dApp

The applications of the paradigm ecosystem are "serverless" (Serverless Application) and are totally decentralized. There are no intermediaries or servers.

They are SPA applications (single-page) in Javascript and Haskell. They contain a library called " blockstack.js. "

The ecosystem of paradigm applications facilitates the exchange of transactions between a financial institution and its end users, and allows the persistence of data in the cloud, in JSON format, so that they can be imported into the organization's own management systems.

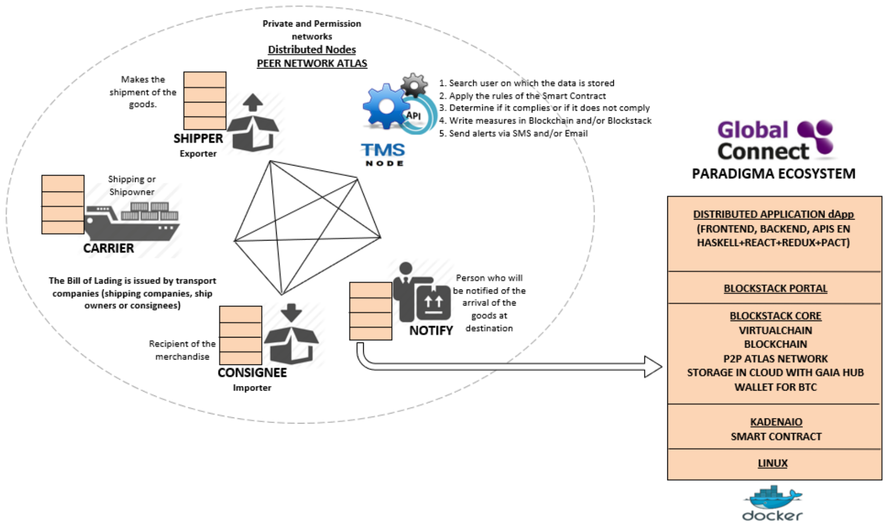
Paradigm applications facilitate, through "Blockstack", manage and authenticate the user, to have access to their resources, perform transactions and allow the storage of data in the cloud. The storage of data is simple and reliable and uses the existing infrastructure in the cloud. Users connect with their Dropbox, Google Drive, One Drive, among others, and the data is synchronized from their local device to the cloud, managed by Gaia Hub.

The identity is controlled by the user and uses the "Blockchain" for the secure management of keys, devices and user names.

The connection of the users with the applications, are anonymous by default and use a specific application key, but their full identity can be revealed at any time. The keys are for signing and encrypting and can be changed.

Through "Blockstack" a decentralized domain name system (DNS) is provided, which allows creating a distributed but closed environment under a domain for a financial organization, that is, where all the actors under the domain participate.

It also allows a decentralized public key distribution system, a registry of applications and also allows the control of user identities.

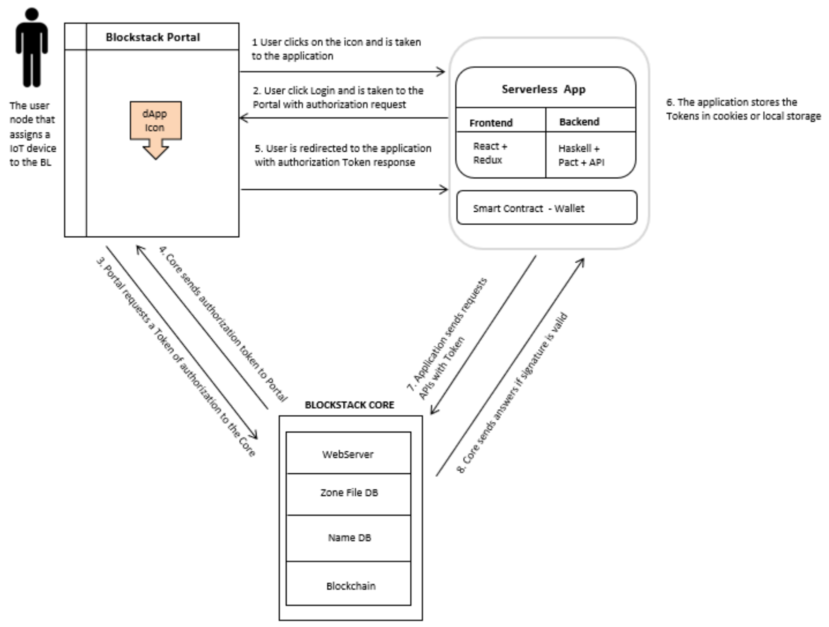

Private and Permission networks
Distributed Nodes
PEER NETWORK ATLAS
Makes the shipment of the goods.
SHIPPER
Exporter
1. Search user on which the data is stored
2. Apply the rules of the Smart Contract
3. Determine if it complies or if it does not comply
4. Write measures in Blockchain and/or Blockstack
5. Send alerts via SMS and/or Email
API
TMS
NODE
Shipping or Shipowner
CARRIER
The Bill of Lading is issued by transport companies (shipping companies, ship owners or consignees)
Person who will be notified of the arrival of the goods at destination
NOTIFY
Recipient of the merchandise
CONSIGNEE
Importer
Global Connect
PARADIGMA ECOSYSTEM
DISTRIBUTED APPLICATION dApp
(FRONTEND, BACKEND, APIS EN HASKELL+REACT+REDUX+PACT)
BLOCKSTACK PORTAL
BLOCKSTACK CORE
VIRTUALCHAIN
BLOCKCHAIN
P2P ATLAS NETWORK
STORAGE IN CLOUD WITH GAIA HUB
WALLET FOR BTC
KADENAIO
SMART CONTRACT
LINUX
docker

Users store and share all the transactions made in the domain, the scripts contain a high level of encryption, the transactions are propagated through the P2P protocol (with the exception of the Smartphone and tablets that must be synchronized -PUSH SYNCHRONIZATION IN BACKGROUND).

Each user has a series of "dockerized" applications installed locally, including:

- A set (one or several) of Web applications (Frontend, Backend) in Javascript and Haskell developed by Paradigma (Dashboard, IoT Devices Initialization Service to link them to the container, container tracking, reports, ...)
- Blockstack Portal
- Blockstack Core
- Wallet
- S.O. Linux

The environment consists of a "Blockstack Portal" as a container for Paradigma applications. The user by pressing on the icon of the application contained in the Portal, is taken to the application. Next, you must authenticate yourself through the login and the application will take you to the Portal to request an authorization token to the Core. The Core sends the token to the Portal, the application stores the token in cookies or localstorage. Then, with the token the application can send API requests to the Core and the latter returns the answers if the signature is valid.


The user node that assigns a IoT device to the BL
Blockstack Portal
dApp Icon
1 User clicks on the icon and is taken to the application
2. User click Login and is taken to the Portal with authorization request
5. User is redirected to the application with authorization Token response
Serverless App
Frontend
Backend
React + Redux
Haskell + Pact + API
Smart Contract - Wallet
6. The application stores the Tokens in cookies or local storage
3. Portal requests a Token of authorization to the Core
4. Core sends authorization token to Portal
7. Application sends requests APIs with Token
8. Core sends answers if signature is valid
BLOCKSTACK CORE
WebServer
Zone File DB
Name DB
Blockchain

3.2.7 Storage of JSON files in the Cloud and Blockchain

The schema of the JSON file that the TMS Node API should create is:

Schema example

```
{
      "DTEvent": string,
       "EnrollmentContainer": string,
       "NumberBillOfLading": string,
       "CallID": string,
       "ContainerCategorySensorId": number,
       "Sensor":
       [
              {
                   "ContainerSensorDescription": string,
                   "ContainerSensorId": number,
                   "FrecuencyNotificationMinutes": number,
                   "Type": string,
                   "Unit": string,
                   "Value": string,
                   "State": string,
                   "LowerThreshold": string,
                   "UpperThreshold": string,
                   "SmartContractRule": string,
                   "DTregistry": string
               }
       ]
}
```

Files in JSON format are stored in Blockchain and in folders in the Cloud.

The data of the sensors, their location, date and time are stored according to the identification number of the container or the identification number of the bill of lading.

Event example

```
    {
          "DTEvent": "2018-10-21T12:34:00:7992",
           "EnrollmentContainer": "BMOU8710633",
           "NumberBillOfLading": "WSZCF09050002",
           "CallID": "9835676546",
           "ContainerCategorySensorId": 34,
           "Sensor":
           [
                  {
                       "ContainerSensorDescription": "PositionsGPS",
                       "ContainerSensorId": 4019,
                       "FrecuencyNotificationMinutes": 2,
                       "Type": "GEOjson",
                       "Units": "",
                       "Value":
                           {
                               "coordinates":
                               [
                                 -0.308595436,
                                 39.43556368
                               ]
                           },
                       "State": "without rule",
                       "LowerThreshold": "",
                       "UpperThreshold": "",
                       "SmartContractRule": "",
                       "DTregistry": "2018-10-21T10:30:44:00Z"
                   },
                   {
                       "ContainerSensorDescription": "Temperature",
                       "ContainerSensorId": 4020,
                       "FrecuencyNotificationMinutes": 2,
                       "Type": "float",
                       "Unit": "°C",
                       "Value": 5,
                       "State": "with rule",
                       "LowerThreshold": -3,
                       "UpperThreshold": 20,
                       "SmartContractRule": "good",
                       "DTregistry": "2018-10-21T10:30:44:00Z"
                   },
                   {
                       "ContainerSensorDescription": "Gases",
                       "ContainerSensorId": 4022,
                       "FrecuencyNotificationMinutes": 2,
                       "Type": "integer",
                       "Unit": "%",
                       "Value": 15,
                       "State": "with rule",
                       "LowerThreshold": 0,
                       "UpperThreshold": 2,
                       "SmartContractRule": "bad",
                       "DTregistry": "2018-10-21T10:30:44:00Z"
                   }
           ]
      }
```

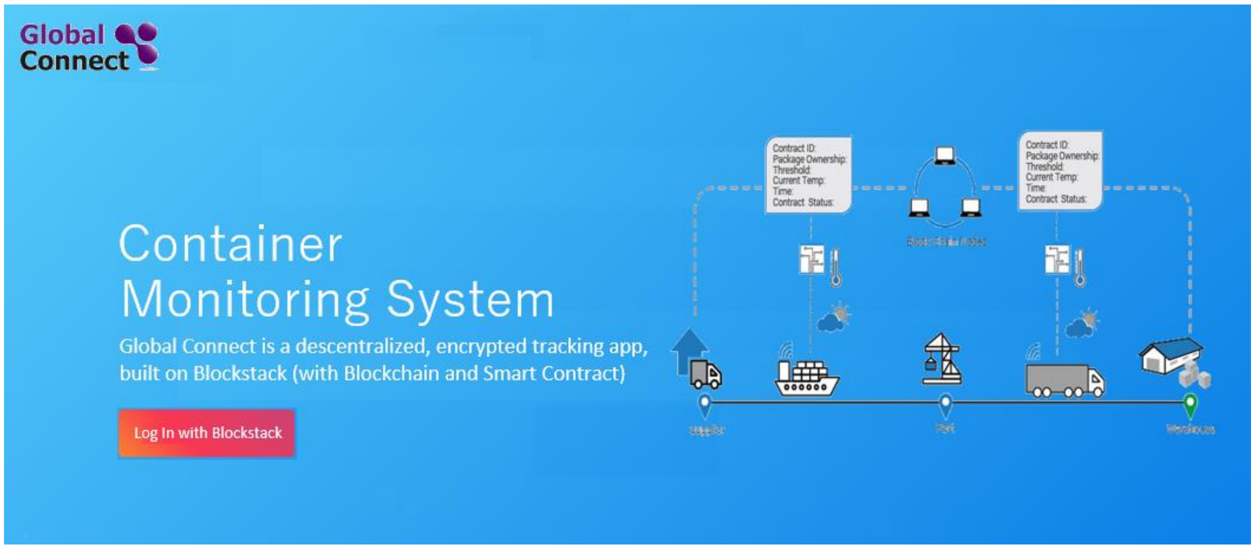
The physical name of the file that is stored in the Cloud contains the data of:

```
EnrollmentContainer + '-' + NumberBillOfLading + '-' + CallID + '-' + ContainerCategorySensorId +
'-' + DTEvent . JSON
```

Example

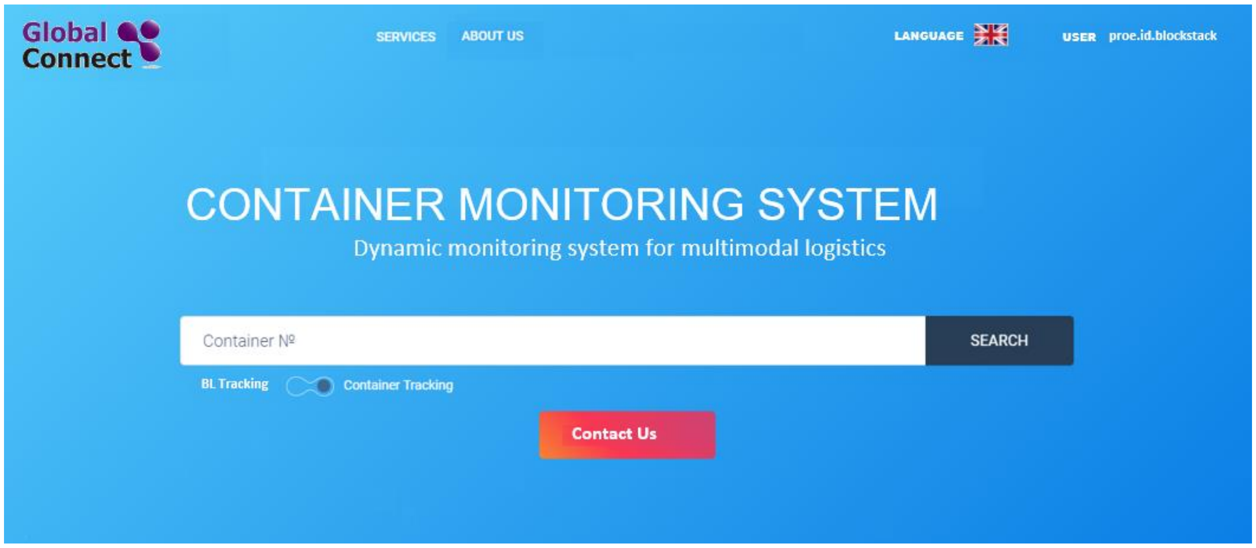
```
BMOU8710633-WSZCF09050002-9835676546-34-2018-10-21T1234007992.json
```

3.2.8 Dashboard



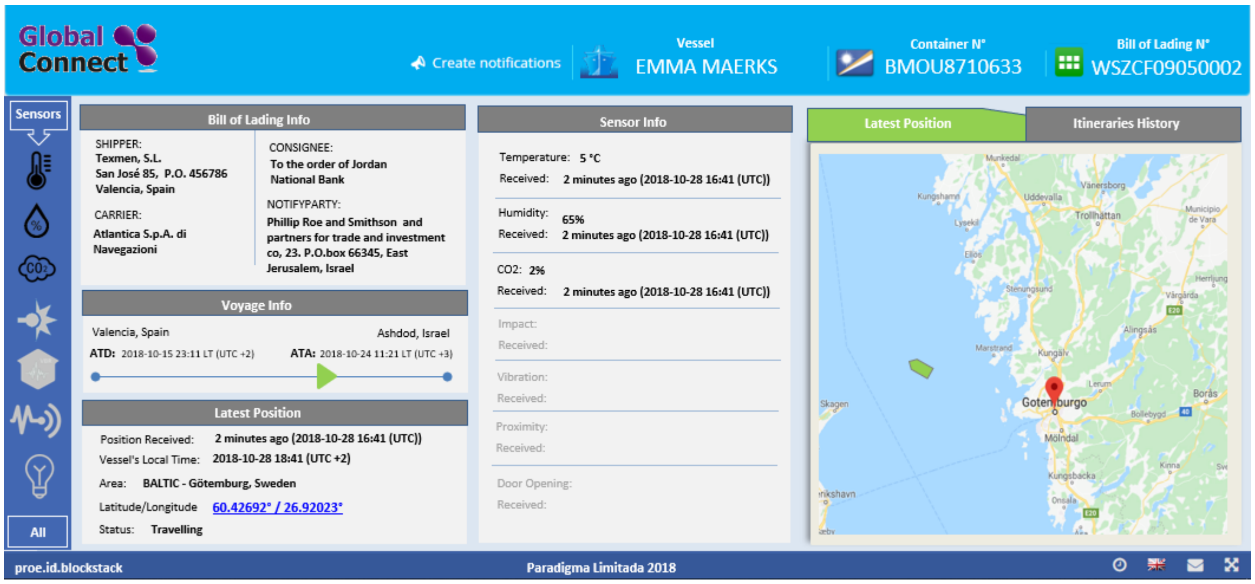
Container Monitoring System

Global Connect is a descentralized, encrypted tracking app, built on Blockstack (with Blockchain and Smart Contract)



CONTAINER MONITORING SYSTEM

Dynamic monitoring system for multimodal logistics



Global Connect

Create notifications

Vessel
EMMA MAERKS

Container N°
BMOU8710633

Bill of Lading N°
WSZCF09050002

Sensors

Bill of Lading Info

SHIPPER:
Texmen, S.L.
San José 85, P.O. 456786
Valencia, Spain

CARRIER:
Atlantica S.p.A. di Navegazioni

CONSIGNEE:
To the order of Jordan National Bank

NOTIFYPARTY:
Phillip Roe and Smithson and partners for trade and investment co, 23. P.O.box 66345, East Jerusalem, Israel

Voyage Info

Valencia, Spain — Ashdod, Israel

ATD: 2018-10-15 23:11 LT (UTC +2) — ATA: 2018-10-24 11:21 LT (UTC +3)

Latest Position

Position Received:	2 minutes ago (2018-10-28 16:41 (UTC))
Vessel's Local Time:	2018-10-28 18:41 (UTC +2)
Area:	BALTIC - Göteburg, Sweden
Latitude/Longitude	60.42692° / 26.92023°
Status:	Travelling

Sensor Info

Temperature: 5 °C
Received: 2 minutes ago (2018-10-28 16:41 (UTC))

Humidity: 65%
Received: 2 minutes ago (2018-10-28 16:41 (UTC))

CO2: 2%
Received: 2 minutes ago (2018-10-28 16:41 (UTC))

Impact:
Received:

Vibration:
Received:

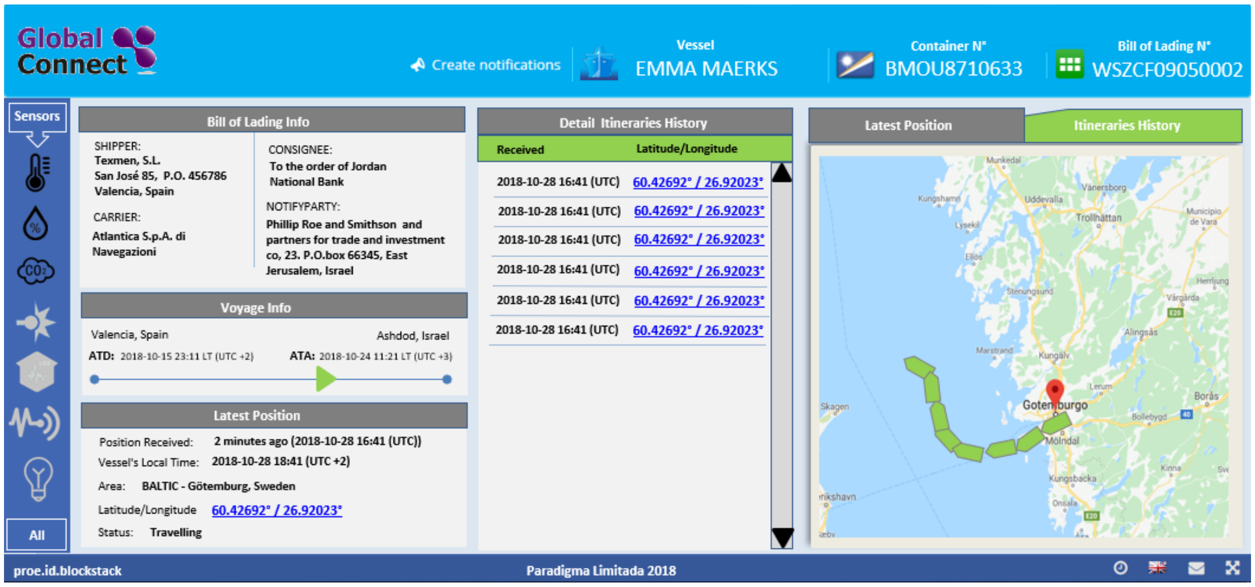
Proximity:
Received:

Door Opening:
Received:

Latest Position | Itineraries History

All

proe.id.blockstack — Paradigma Limitada 2018



Global Connect

Create notifications

Vessel
EMMA MAERKS

Container N°
BMOU8710633

Bill of Lading N°
WSZCF09050002

Sensors

Bill of Lading Info

SHIPPER:
Texmen, S.L.
San José 85, P.O. 456786
Valencia, Spain

CARRIER:
Atlantica S.p.A. di Navegazioni

CONSIGNEE:
To the order of Jordan National Bank

NOTIFYPARTY:
Phillip Roe and Smithson and partners for trade and investment co, 23. P.O.box 66345, East Jerusalem, Israel

Voyage Info

Valencia, Spain — Ashdod, Israel

ATD: 2018-10-15 23:11 LT (UTC +2) — ATA: 2018-10-24 11:21 LT (UTC +3)

Latest Position

Position Received:	2 minutes ago (2018-10-28 16:41 (UTC))
Vessel's Local Time:	2018-10-28 18:41 (UTC +2)
Area:	BALTIC - Göteburg, Sweden
Latitude/Longitude	60.42692° / 26.92023°
Status:	Travelling

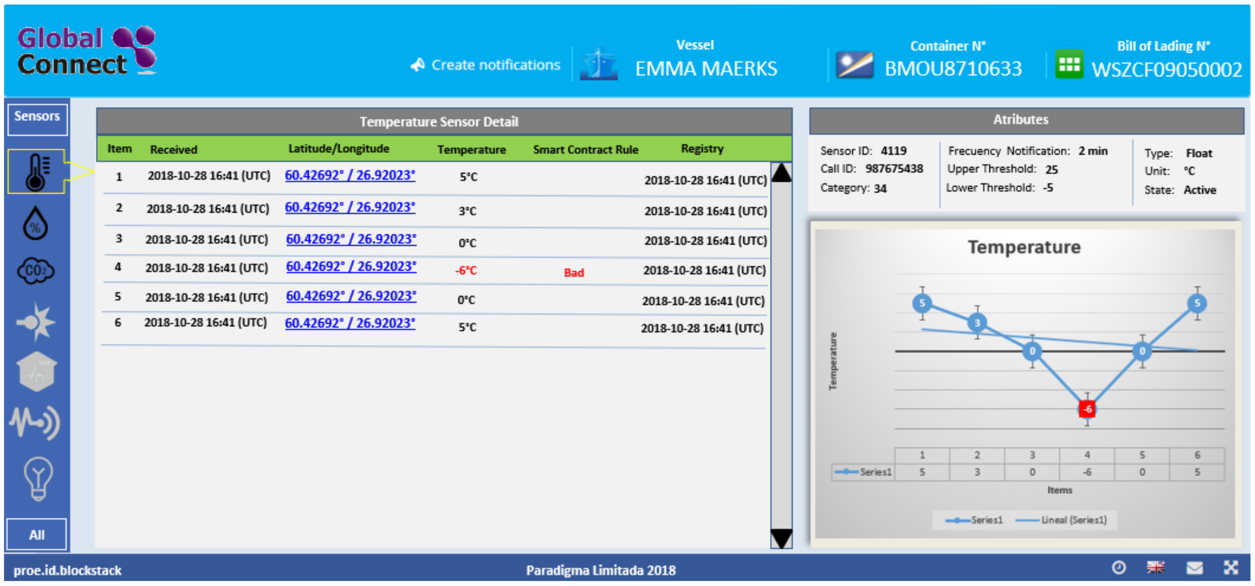
Detail Itineraries History

Received	Latitude/Longitude
2018-10-28 16:41 (UTC)	60.42692° / 26.92023°
2018-10-28 16:41 (UTC)	60.42692° / 26.92023°
2018-10-28 16:41 (UTC)	60.42692° / 26.92023°
2018-10-28 16:41 (UTC)	60.42692° / 26.92023°
2018-10-28 16:41 (UTC)	60.42692° / 26.92023°
2018-10-28 16:41 (UTC)	60.42692° / 26.92023°

Latest Position | Itineraries History

All

proe.id.blockstack — Paradigma Limitada 2018



Vessel EMMA MAERKS | Container N° BMOU8710633 | Bill of Lading N° WSZCF09050002

Temperature Sensor Detail

Item	Received	Latitude/Longitude	Temperature	Smart Contract Rule	Registry
1	2018-10-28 16:41 (UTC)	60.42692° / 26.92023°	5°C		2018-10-28 16:41 (UTC)
2	2018-10-28 16:41 (UTC)	60.42692° / 26.92023°	3°C		2018-10-28 16:41 (UTC)
3	2018-10-28 16:41 (UTC)	60.42692° / 26.92023°	0°C		2018-10-28 16:41 (UTC)
4	2018-10-28 16:41 (UTC)	60.42692° / 26.92023°	-6°C	Bad	2018-10-28 16:41 (UTC)
5	2018-10-28 16:41 (UTC)	60.42692° / 26.92023°	0°C		2018-10-28 16:41 (UTC)
6	2018-10-28 16:41 (UTC)	60.42692° / 26.92023°	5°C		2018-10-28 16:41 (UTC)

Atributes

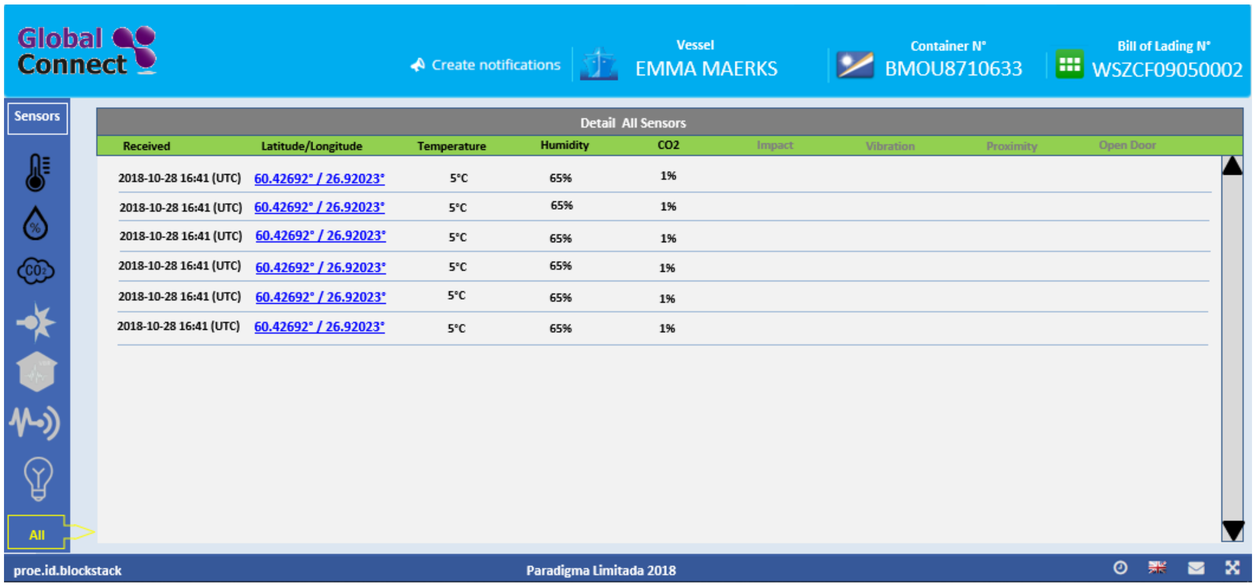
Sensor ID: 4119
Call ID: 987675438
Category: 34

Frecuency Notification: 2 min
Upper Threshold: 25
Lower Threshold: -5

Type: Float
Unit: °C
State: Active

	1	2	3	4	5	6
Series1	5	3	0	-6	0	5

proe.id.blockstack   Paradigma Limitada 2018

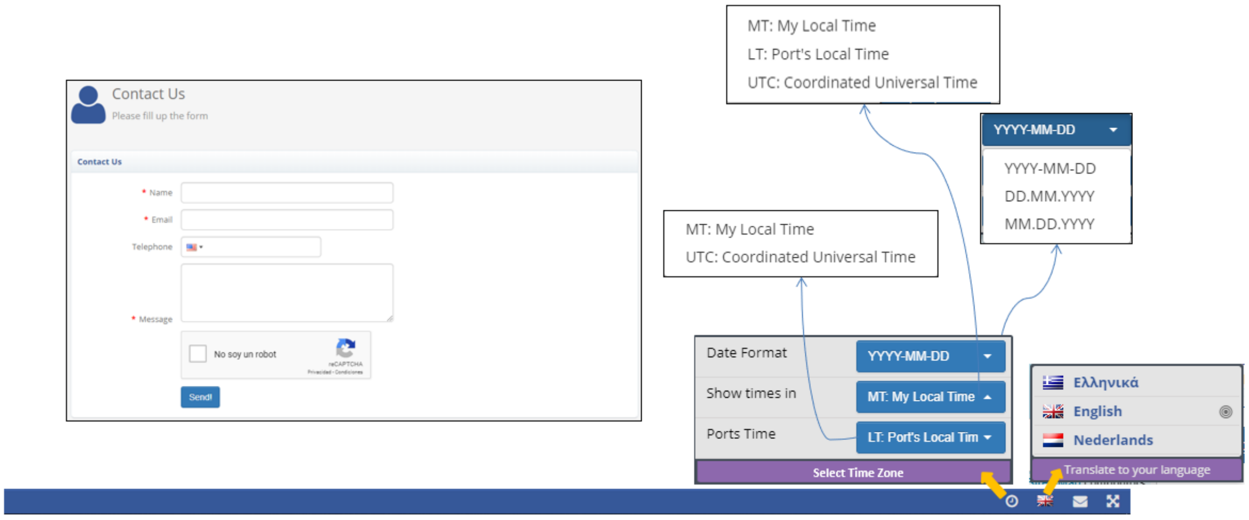


Vessel EMMA MAERKS | Container N° BMOU8710633 | Bill of Lading N° WSZCF09050002

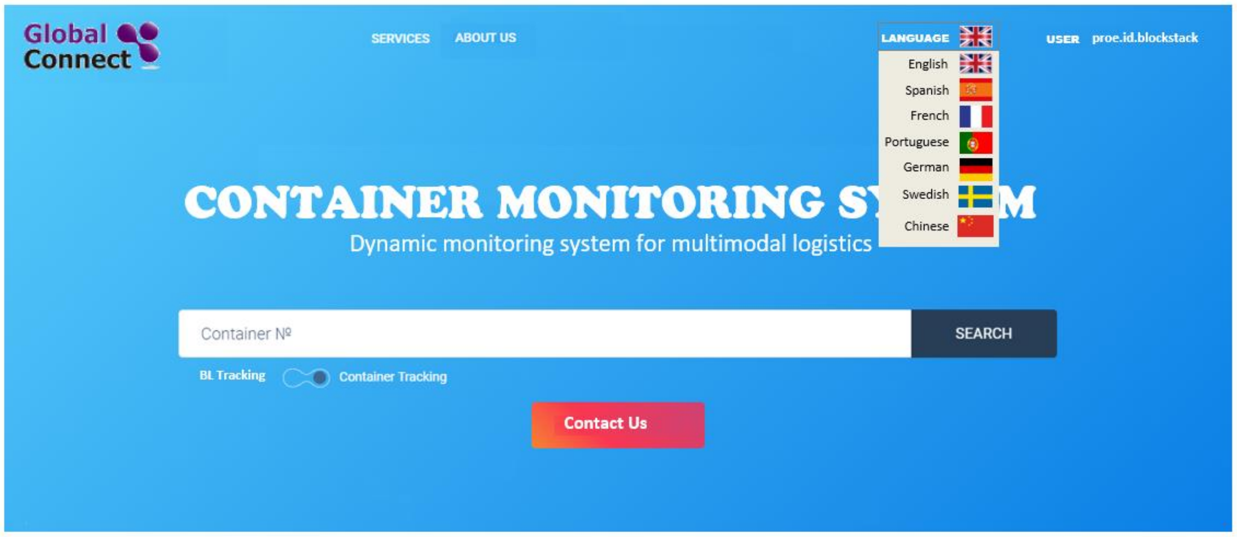
Detail All Sensors

Received	Latitude/Longitude	Temperature	Humidity	CO2	Impact	Vibration	Proximity	Open Door
2018-10-28 16:41 (UTC)	60.42692° / 26.92023°	5°C	65%	1%				
2018-10-28 16:41 (UTC)	60.42692° / 26.92023°	5°C	65%	1%				
2018-10-28 16:41 (UTC)	60.42692° / 26.92023°	5°C	65%	1%				
2018-10-28 16:41 (UTC)	60.42692° / 26.92023°	5°C	65%	1%				
2018-10-28 16:41 (UTC)	60.42692° / 26.92023°	5°C	65%	1%				
2018-10-28 16:41 (UTC)	60.42692° / 26.92023°	5°C	65%	1%				

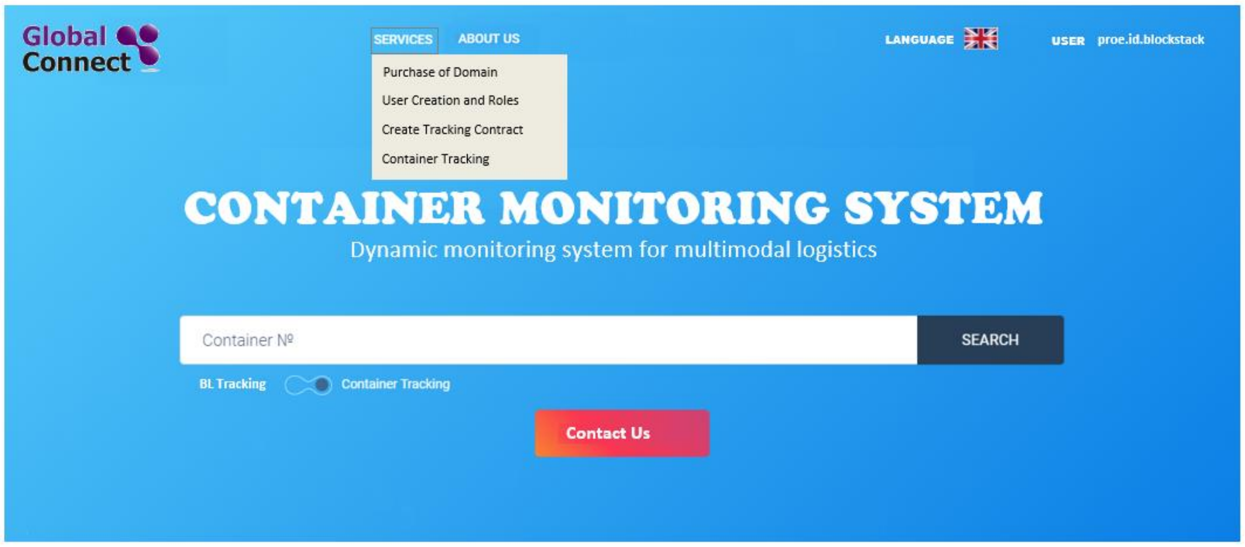
proe.id.blockstack   Paradigma Limitada 2018

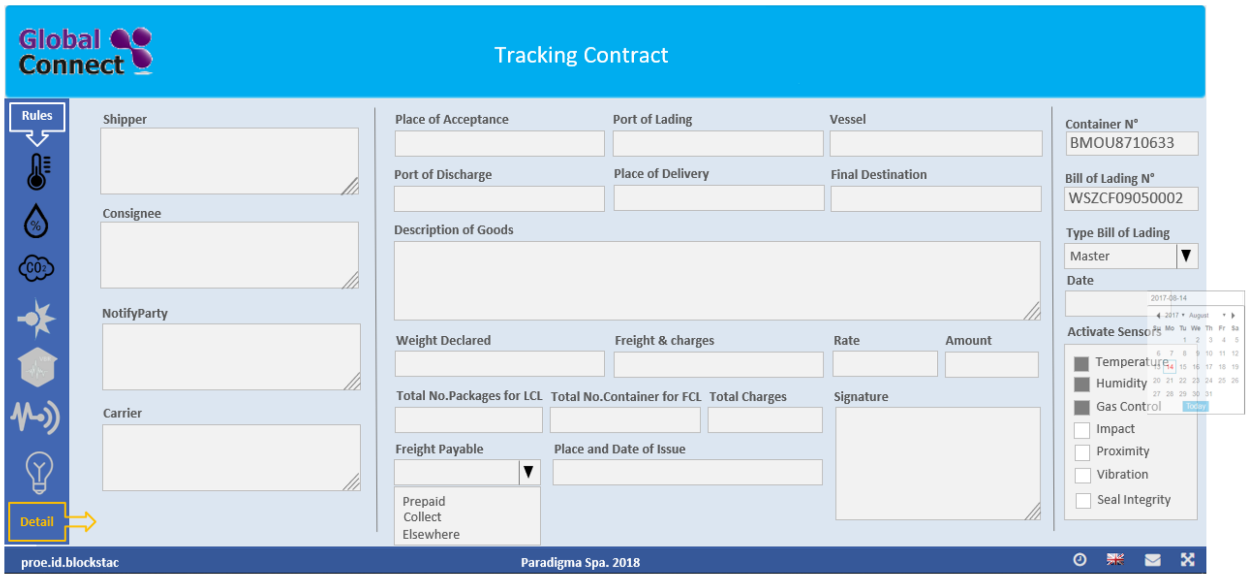


3.2.9 Languages



3.2.10 Smart Contract Mantenaince





Shipper

Consignee

NotifyParty

Carrier

Place of Acceptance | Port of Lading | Vessel

Port of Discharge | Place of Delivery | Final Destination

Description of Goods

Weight Declared | Freight & charges | Rate | Amount

Total No.Packages for LCL | Total No.Container for FCL | Total Charges | Signature

Freight Payable: Prepaid / Collect / Elsewhere

Place and Date of Issue

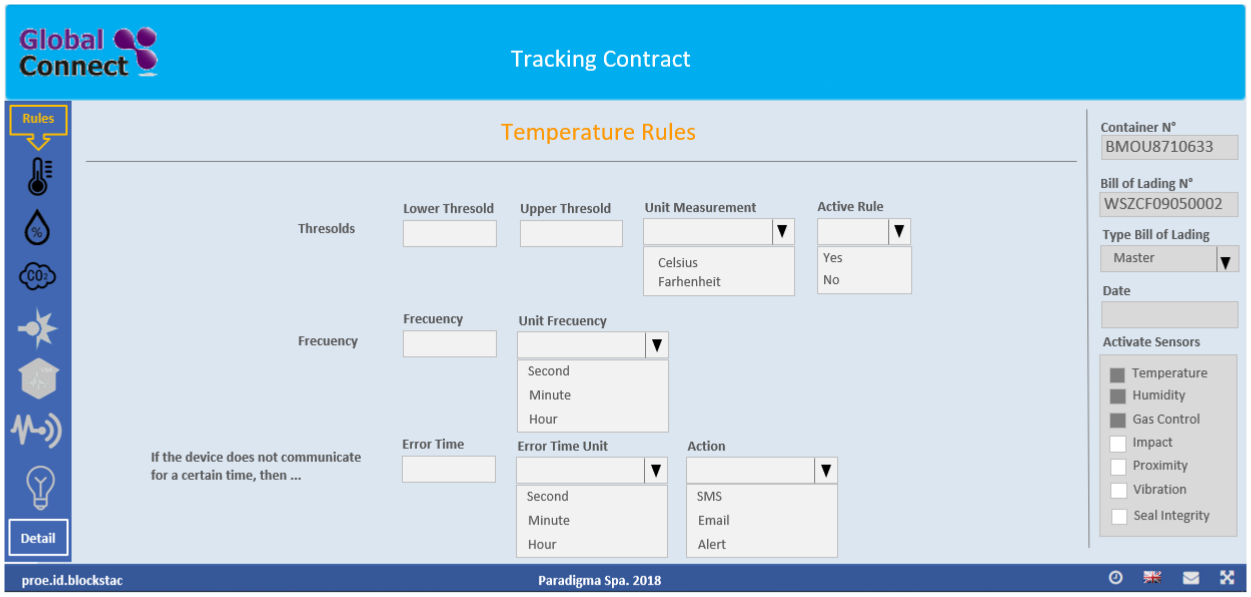
Container N°: BMOU8710633

Bill of Lading N°: WSZCF09050002

Type Bill of Lading: Master

Date: 2017-08-14

Activate Sensors
- ■ Temperature
- ■ Humidity
- ■ Gas Control
- ☐ Impact
- ☐ Proximity
- ☐ Vibration
- ☐ Seal Integrity



Tracking Contract

Temperature Rules

Thresolds — Lower Thresold; Upper Thresold; Unit Measurement: Celsius, Farhenheit; Active Rule: Yes, No

Frecuency — Frecuency; Unit Frecuency: Second, Minute, Hour

If the device does not communicate for a certain time, then ... — Error Time; Error Time Unit: Second, Minute, Hour; Action: SMS, Email, Alert

Container N°: BMOU8710633

Bill of Lading N°: WSZCF09050002

Type Bill of Lading: Master

Date:

Activate Sensors

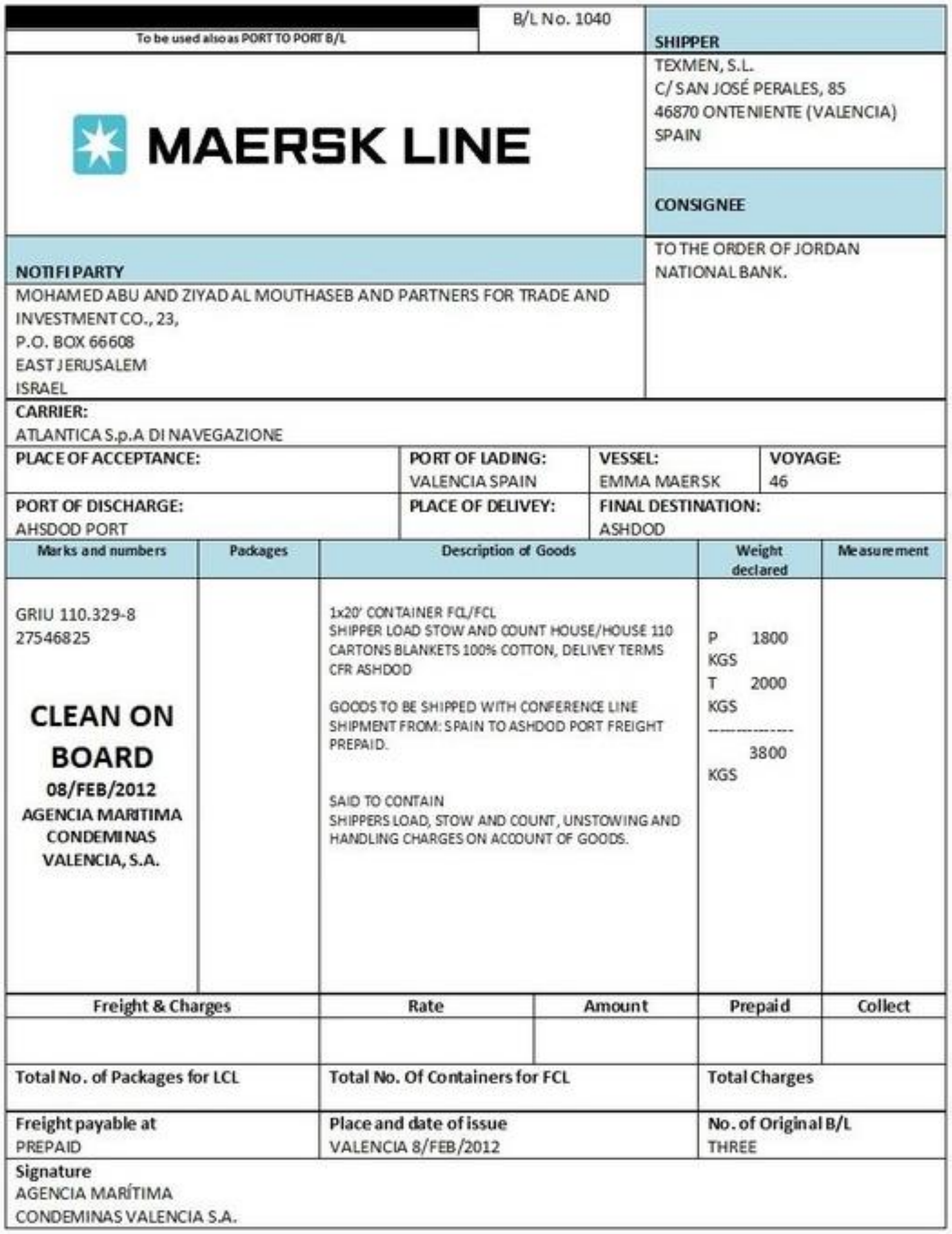
- ■ Temperature
- ■ Humidity
- ■ Gas Control
- ☐ Impact
- ☐ Proximity
- ☐ Vibration
- ☐ Seal Integrity

proe.id.blockstac   Paradigma Spa. 2018

3.2.11 Reports and Data Analisys

3.2.10 Data Model

❖ *Bill Of Lading*



To be used also as PORT TO PORT B/L

B/L No. 1040

MAERSK LINE

SHIPPER
TEXMEN, S.L.
C/ SAN JOSÉ PERALES, 85
46870 ONTENIENTE (VALENCIA)
SPAIN

CONSIGNEE
TO THE ORDER OF JORDAN NATIONAL BANK.

NOTIFI PARTY
MOHAMED ABU AND ZIYAD AL MOUTHASEB AND PARTNERS FOR TRADE AND INVESTMENT CO., 23,
P.O. BOX 66608
EAST JERUSALEM
ISRAEL

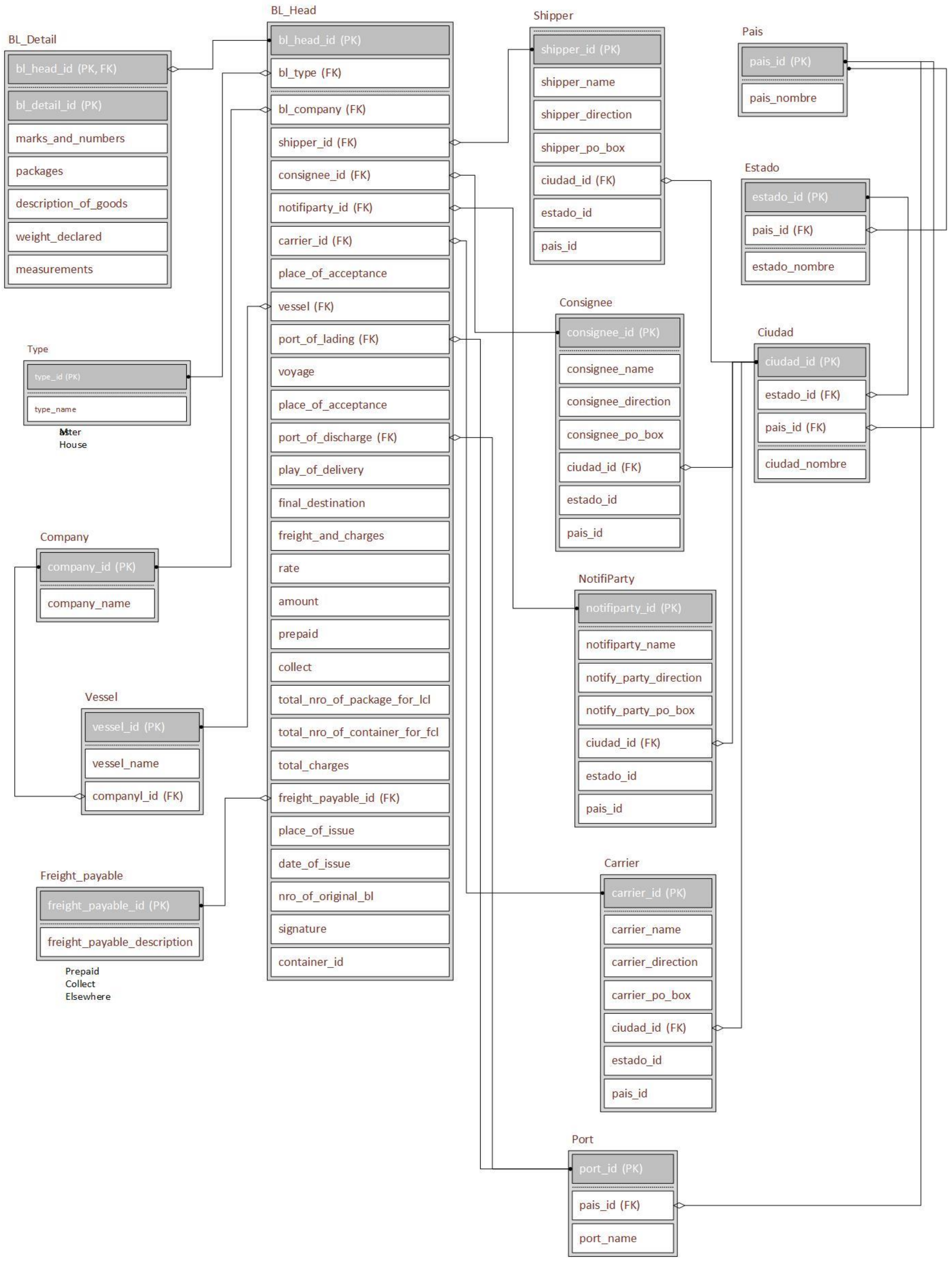
CARRIER:
ATLANTICA S.p.A DI NAVEGAZIONE

PLACE OF ACCEPTANCE:	**PORT OF LADING:** VALENCIA SPAIN	**VESSEL:** EMMA MAERSK	**VOYAGE:** 46
PORT OF DISCHARGE: AHSDOD PORT	**PLACE OF DELIVEY:**	**FINAL DESTINATION:** ASHDOD	

Marks and numbers	Packages	Description of Goods	Weight declared	Measurement
GRIU 110.329-8 27546825 **CLEAN ON BOARD** **08/FEB/2012** **AGENCIA MARITIMA CONDEMINAS VALENCIA, S.A.**		1x20' CONTAINER FCL/FCL SHIPPER LOAD STOW AND COUNT HOUSE/HOUSE 110 CARTONS BLANKETS 100% COTTON, DELIVEY TERMS CFR ASHDOD GOODS TO BE SHIPPED WITH CONFERENCE LINE SHIPMENT FROM: SPAIN TO ASHDOD PORT FREIGHT PREPAID. SAID TO CONTAIN SHIPPERS LOAD, STOW AND COUNT, UNSTOWING AND HANDLING CHARGES ON ACCOUNT OF GOODS.	P 1800 KGS T 2000 KGS --------------- 3800 KGS	

Freight & Charges	Rate	Amount	Prepaid	Collect
Total No. of Packages for LCL	Total No. Of Containers for FCL		Total Charges	
Freight payable at PREPAID	Place and date of issue VALENCIA 8/FEB/2012		No. of Original B/L THREE	

Signature
AGENCIA MARÍTIMA
CONDEMINAS VALENCIA S.A.


BL_Detail
bl_head_id (PK, FK)
bl_detail_id (PK)
marks_and_numbers
packages
description_of_goods
weight_declared
measurements
Type
type_id (PK)
type_name
Master
House
Company
company_id (PK)
company_name
Vessel
vessel_id (PK)
vessel_name
companyl_id (FK)
Freight_payable
freight_payable_id (PK)
freight_payable_description
Prepaid
Collect
Elsewhere
BL_Head
bl_head_id (PK)
bl_type (FK)
bl_company (FK)
shipper_id (FK)
consignee_id (FK)
notifiparty_id (FK)
carrier_id (FK)
place_of_acceptance
vessel (FK)
port_of_lading (FK)
voyage
place_of_acceptance
port_of_discharge (FK)
play_of_delivery
final_destination
freight_and_charges
rate
amount
prepaid
collect
total_nro_of_package_for_lcl
total_nro_of_container_for_fcl
total_charges
freight_payable_id (FK)
place_of_issue
date_of_issue
nro_of_original_bl
signature
container_id
Shipper
shipper_id (PK)
shipper_name
shipper_direction
shipper_po_box
ciudad_id (FK)
estado_id
pais_id
Consignee
consignee_id (PK)
consignee_name
consignee_direction
consignee_po_box
ciudad_id (FK)
estado_id
pais_id
NotifiParty
notifiparty_id (PK)
notifiparty_name
notify_party_direction
notify_party_po_box
ciudad_id (FK)
estado_id
pais_id
Carrier
carrier_id (PK)
carrier_name
carrier_direction
carrier_po_box
ciudad_id (FK)
estado_id
pais_id
Port
port_id (PK)
pais_id (FK)
port_name
Pais
pais_id (PK)
pais_nombre
Estado
estado_id (PK)
pais_id (FK)
estado_nombre
Ciudad
ciudad_id (PK)
estado_id (FK)
pais_id (FK)
ciudad_nombre

The schema of the JSON file is:

Schema

```
    {
          "BL_head": string,
          "BL_type":
                   {
                         "type_id": string,
                         "type_name": string
                   },
          "BL_company":
                  {
                         "company_id": string,
                         "company_name": string
                  },
          "BL_shipper":
                  {
                         "shipper_id": string,
                         "shipper_name": string,
                         "shipper_direction": string,
                         "shipper_po_box": string,
                         "city":
                              {
                                 "city_id": string,
                                 "city_name": string,
                                 "region_state":
                                      {
                                         "region_state_id": string,
                                         "region_state_name": string,
                                         "country":
                                              {
                                                 "contry_id": string,
                                                 "country_name": string
                                              }
                                      }
                              }

                  },
          "BL_consignee":
                  {
                         "consignee_id": string,
                         "consignee_name": string,
                         "consignee_direction": string,
                         "consignee_po_box": string,
                         "city":
                              {
                                 "city_id": string,
                                 "city_name": string,
                                 "region_state":
                                      {
                                         "region_state_id": string,
                                         "region_state_name": string,
                                         "country":
                                              {
                                                 "contry_id": string,
                                                 "country_name": string
                                              }
                                      }
                              }

                  },
          "BL_notifyparty":
                  {
                         "notifyparty_id": string,
                         "notifyparty_name": string,
                         "notifyparty_direction": string,
                         "notifyparty_po_box": string,
                         "city":
                              {
```

```
                    "city_id": string,
                    "city_name": string,
                    "region_state":
                         {
                           "region_state_id": string,
                           "region_state_name": string,
                           "country":
                                {
                                  "contry_id": string,
                                  "country_name": string
                                 }
                         }
                  }

          },
 "BL_carrier":
          {
               "carrier_id": string,
               "carrier_name": string,
               "carrier_direction": string,
               "carrier_po_box": string,
               "city":
                    {
                       "city_id": string,
                       "city_name": string,
                       "region_state":
                            {
                              "region_state_id": string,
                              "region_state_name": string,
                              "country":
                                   {
                                     "contry_id": string,
                                     "country_name": string
                                    }
                            }
                     }

          },

  "place_of_acceptance": string,
  "Vessel":
          {
               "vessel_id": string,
               "vessel_name": string
          },
  "Port_of_lading":
          {
               "port_id": string,
               "port_name": string,
               "country":
                   {
                      "country_id": string,
                      "country_name": string
                   }
          },
  "voyage": string,
  "place_of_aceptance": string,
  "Port_of_discharge":
          {
               "port_id": string,
               "port_name": string,
               "country":
                   {
                      "country_id": string,
                      "country_name": string
                   }
          },
  "play_of_delivery": string,
  "final_destination": string,
  "freight_and_charges": string,
  "rate": string,
  "amount": string,
          "prepaid": string,
          "collect": string,
          "total_nro_of_package_for_lcl": string,
          "total_nro_of_container_for_fcl": string,
          "total_charges": string,
          "Freight_payable":
                  {
                      "freight_payable_id": string,
                      "freight_payable_name": string
                  },
          "place_of_issue": string,
          "date_of_issue": string,
          "nro_of_original_bl": string,
          "signature": string,
          "container_id": string,
          "BL_Detail":
           [
                  {
                      "BL_detail_id": string,
                      "marks_and_number": string,
                      "packages": string,
                      "description_of_goods": string,
                      "weight_declared": string,
                      "measurements": string
                  }
           ]

}
```

Files in JSON format are stored in the Cloud.

Example

```
    {
         "BL_head": "WSZCF09050002",
         "BL_type":
                  {
                      "type_id": "1",
                      "type_name": "Master"
                  },
         "BL_company":
                 {
                      "company_id": "23",
                      "company_name": "Maerks Line"
                 },
         "BL_shipper":
                 {
                      "shipper_id": "1034",
                      "shipper_name": "Texmen S.L.",
                      "shipper_direction": "Calle San José Perales 85",
                      "shipper_po_box": "46870",
                      "city":
                           {
                              "city_id": "246",
                              "city_name": "Valencia",
                              "region_state":
                                   {
                                      "region_state_id": "62",
                                      "region_state_name": "Pais Valenciano",
                                      "country":
                                           {
                                              "country_id": "34",
                                              "country_name": "Spain"
                                           }
                                   }
                           }

                 },
         "BL_consignee":
                 {
```

```
                "consignee_id": "766",
                "consignee_name": "The Jordan National Bank",
                "consignee_direction": "",
                "consignee_po_box": "",
                "city":
                     {
                       "city_id": "",
                       "city_name": "",
                       "region_state":
                            {
                              "region_state_id": "",
                              "region_state_namee": "",
                              "country":
                                   {
                                     "country_id": "",
                                     "country_name": ""
                                   }
                            }
                     }

          },
  "BL_notifyparty":
          {
                "notifyparty_id": "49",
                "notifyparty_name": "Phillip Roe and Smithson and Partners for Trade
                                     And Investment CO",
                "notifyparty_direction": "Calle Ben Gurion 23",
                "notifyparty_po_box": "66608",
                "city":
                     {
                       "city_id": "768",
                       "city_name": "Jerusalem",
                       "region_state":
                            {
                              "region_state_id": "7689",
                              "region_state_name": "East Jerusalem",
                              "country":
                                   {
                                     "country_id": "37",
                                     "country_name": "Israel"
                                   }
                            }
                     }

          },
  "BL_carrier":
          {
                "carrier_id": "57",
                "carrier_name": "Atlantica S.p.A.",
                "carrier_direction": "",
                "carrier_po_box": "",
                "city":
                     {
                       "city_id": "",
                       "city_name": "",
                       "region_state":
                            {
                              "region_state_id": "",
                              "region_state_name": "",
                              "country":
                                   {
                                     "country_id": "",
                                     "country_nombre": ""
                                   }
                            }
                     }

          },

   "place_of_acceptance": "",
   "Vessel":
          {
            "vessel_id": "764565",
            "vessel_name": "Emma Maerks"
        },
"Port_of_lading":
        {
            "port_id": "65368",
            "port_name": "Valencia",
            "country":
                {
                  "country_id": "34",
                  "country_name": "Spain"
                }
        },
"voyage": "46",
"place_of_aceptance": "",
"Port_of_discharge":
        {
            "port_id": "445",
            "port_name": "Ashdod",
            "country":
                {
                  "country_id": "37",
                  "country_name": "Israel"
                }
        },
"play_of_delivery": "",
"final_destination": "Ashdod",
"freight_and_charges": ",
"rate": "",
"amount": "",
"prepaid": "",
"collect": "",
"total_nro_of_package_for_lcl": "",
"total_nro_of_container_for_fcl": "",
"total_charges": "",
"Freight_payable":
        {
            "freight_payable_id": "1",
            "freight_payable_name": "Prepaid"
        },
"place_of_issue": "Valencia",
"date_of_issue": "08/02/2018",
"nro_of_original_bl": "Three",
"signature": "Agencia Maritima Condeminas Valencia S.A.",
"container_id": "BMOU8710633",
"BL_Detail":
 [
        {
            "BL_detail_id": "1",
            "marks_and_number": "",
            "packages": "20",
            "description_of_goods": "1x20' CONTAINER FCL/FCL SHIPPER LOAD STOW AND
                                     COUNT HOUSE/HOUSE 110 CARTONS BLANKETS 100%
                                     COTTON, DELIVEY TERMS CFR ASHDOD",
            "weight_declared": "1800 KGS",
            "measurements": ""
        },
        {
            "BL_detail_id": "2",
            "marks_and_number": "",
            "packages": "35",
            "description_of_goods": "GOCOS TO BE SHIPPED WITH CONFERENCE LINE SHIPMENT
                                     FROM: SPAIN TO ASHDOD PORT FREIHT PREPAID",
            "weight_declared": "2000 KGS",
            "measurements": ""
        },
        {
            "BL_detail_id": "3",
            "marks_and_number": "",
            "packages": "14",
            "description_of_goods": "SAID TO CONTAIN SHIPPERS LOAD, STOW AND COUNT,
                                     UNSTOWING AND HANDLING CHARGES ON ACCOUNT OF
```

```
                                                    GOODS.",
                      "weight_declared": "",
                      "measurements": ""
                  }
              ]

}
```

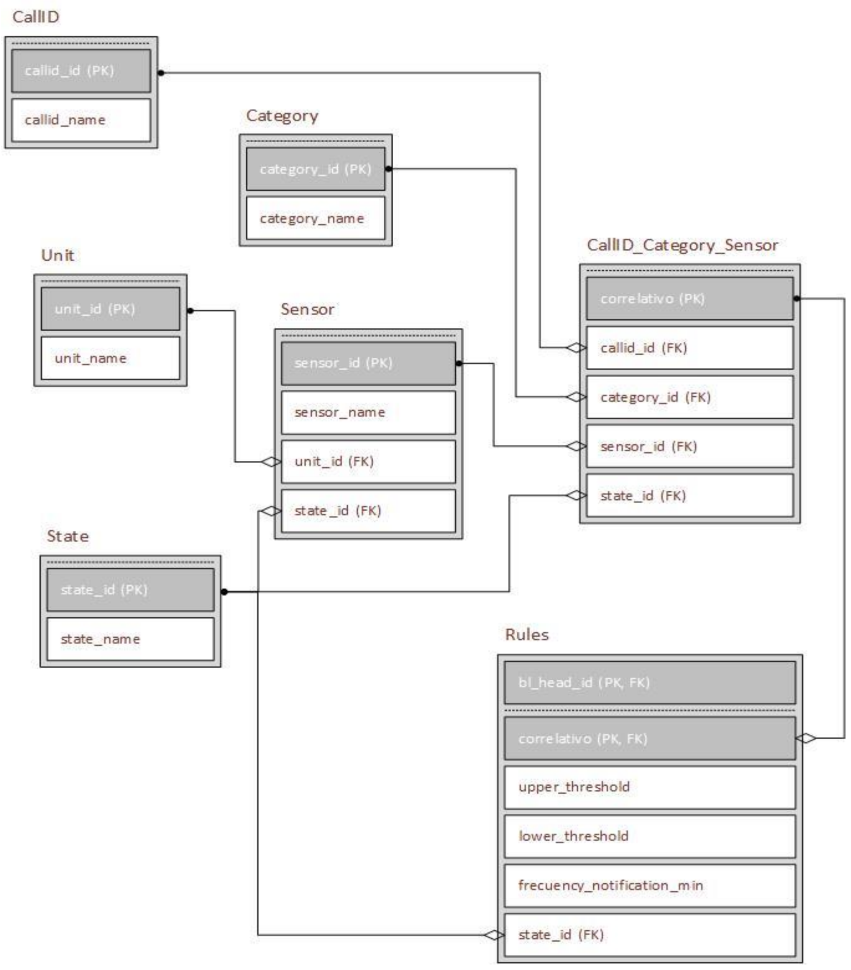
The physical name of the file that is stored in the Cloud contains the data of:

```
"BL_" + NumberBillOfLading . JSON
```

Example

```
BL_WSZCF09050002.json
```

❖ *Smart Contract Rules*



The schema of the JSON file is:

Schema

```
{
   "bl_head":
       {
          "bl_head_id": string,
          "container_id": string
       },
    "Rules":
    [
       {
          "Correlative":
                  {
                      "correlative_id": string,
                      "Callid":
                         {
                            "callid_id": string,
                            "callid_name": string
                         },
                      "Category":
                         {
                            "category_id": string,
                            "category_name": string
                         },
                      "Sensor":
                         {
                            "sensor_id": string,
                            "sensor_name": string,
                            "Unit":
                               {
                                 "unit_id": string,
                                 "unit_name": string
                               },
                            "State":
                               {
                                 "state_id": string,
                                 "state_name": string
                               }
                         },
                       "State2":
                         {
                            "state_id2": string,
                            "state_name2": string
                         }
                   },
          "upper_thresold": string,
          "lower_thresold": string,
          "frecuency_notification_min": string
          "State3":
              {
                 "state_id3": string,
                 "state_name3": string
              }
       }

     ]

}
```

Files in JSON format are stored in the Distributed Storage (Cloud).

Example

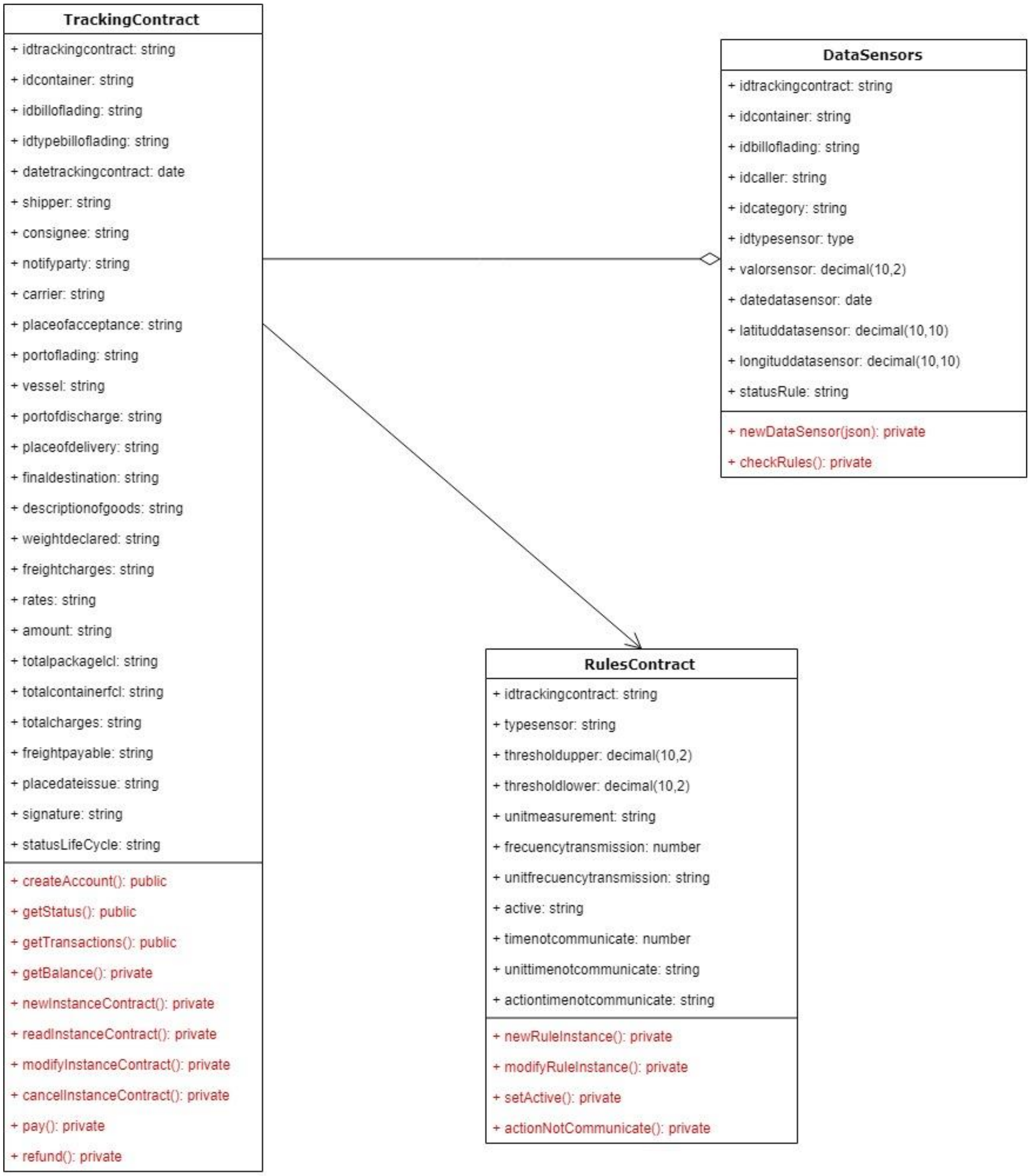
```
{
   "bl_head":
       {
          "bl_head_id": "WSZCF09050002",
          "container_id": "BMOU8710633"
       },
    "Rules":
       {
          "Correlative":
                   {
                       "correlative_id": "1",
                       "Callid":
                          {
                             "callid_id": "89403933",
                             "callid_name": "89403933"
                          },
                       "Category":
                          {
                             "category_id": "34",
                             "category_name": "34"
                          },
                       "Sensor":
                          {
                             "sensor_id": "4219",
                             "sensor_name": "Temperature",
                             "Unit":
                                {
                                  "unit_id": "45",
                                  "unit_name": "°C"
                                },
                             "State":
                                {
                                  "state_id": "10",
                                  "state_name": "Active"
                                }
                          },
                        "State2":
                          {
                             "state_id2": "10",
                             "state_name2": "Active"
                          }
                    },
          "upper_thresold": "25",
          "lower_thresold": "-5",
          "frecuency_notification_min": "2"
          "State3":
               {
                  "state_id3": "10",
                  "state_name3": "Active"
               }
        },
       {
          "Correlativo":
                   {
                       "correlativo_id": "2",
                       "Callid":
                          {
                             "callid_id": "89403934",
                             "callid_name": "89403934"
                          },
                       "Category":
                          {
                             "category_id": "35",
                             "category_name": "35"
                          },
                       "Sensor":
                          {
                             "sensor_id": "4220",
                             "sensor_name": "Humidity",
                             "Unit":
                                {
                                  "unit_id": "46",
                                  "unit_name": "%"
                                },
                             "State":
                                {
                                  "state_id": "10",
                                  "state_name": "Active"
                                }
                          },
                        "State2":
                          {
                             "state_id2": "10",
                             "state_name2": "Active"
                          }
                    },
           "upper_thresold": "100",
           "lower_thresold": "0",
           "frecuency_notification_min": "2"
           "State3":
               {
                  "state_id3": "10",
                  "state_name3": "Active"
               }
        }
}
```

The physical name of the file that is stored in the Cloud contains the data of:

```
"BL_" + "RULES_" + NumberBillOfLading . JSON
```

Example

```
BL_RULES_WSZCF09050002.json
```


TrackingContract
+ idtrackingcontract: string
+ idcontainer: string
+ idbillofladiing: string
+ idtypebillofladiing: string
+ datetrackingcontract: date
+ shipper: string
+ consignee: string
+ notifyparty: string
+ carrier: string
+ placeofacceptance: string
+ portofladiing: string
+ vessel: string
+ portofdischarge: string
+ placeofdelivery: string
+ finaldestination: string
+ descriptionofgoods: string
+ weightdeclared: string
+ freightcharges: string
+ rates: string
+ amount: string
+ totalpackagelcl: string
+ totalcontainerfcl: string
+ totalcharges: string
+ freightpayable: string
+ placedateissue: string
+ signature: string
+ statusLifeCycle: string
+ createAccount(): public
+ getStatus(): public
+ getTransactions(): public
+ getBalance(): private
+ newInstanceContract(): private
+ readInstanceContract(): private
+ modifyInstanceContract(): private
+ cancelInstanceContract(): private
+ pay(): private
+ refund(): private
DataSensors
+ idtrackingcontract: string
+ idcontainer: string
+ idbillofladiing: string
+ idcaller: string
+ idcategory: string
+ idtypesensor: type
+ valorsensor: decimal(10,2)
+ datedatasensor: date
+ latituddatasensor: decimal(10,10)
+ longituddatasensor: decimal(10,10)
+ statusRule: string
+ newDataSensor(json): private
+ checkRules(): private
RulesContract
+ idtrackingcontract: string
+ typesensor: string
+ thresholdupper: decimal(10,2)
+ thresholdlower: decimal(10,2)
+ unitmeasurement: string
+ frecuencytransmission: number
+ unitfrecuencytransmission: string
+ active: string
+ timenotcommunicate: number
+ unittimenotcommunicate: string
+ actiontimenotcommunicate: string
+ newRuleInstance(): private
+ modifyRuleInstance(): private
+ setActive(): private
+ actionNotCommunicate(): private

4. GLOSSARY OF TÉRMS

6loWPAN. IPv6 over Low Power Wireless Personal Area Networks

API. Application Programming Interface

ATA. Actual Time of Arrival (at Port of Destination)

ATD. Actual Time of Departure (from Port of Origin)

BLE. Bluetooh Low Energy

CEP. Complex Event Processing

CO2. Carbon Dioxide

CORS. Cross-Origin Resource Sharing

CPS. Cyber Physical System

CTS. Container Tracking Service

DAG. Directed Acyclic Graph

DLT. Distributed Ledger Technologies

DSNS. Domain Sensor Name Server

EPL. Event Processing Language

ETA. Estimated Time of Arrival (at Port of Destination)

ETD. Estimated Time of Departure (from Port of Origin)

GPS. Global Positioning System

HMI. Human Machine Interface

HTTP. Hypertext Transfer Protocol

HTML. HyperText Markup Language

IANA. Internet Assigned Numbers Authority

ICO. Initial Coin Offering

IDE. Integrated Development Environment

IEEE. Institute of Electrical and Electronics Engineers

IETF. Internet Engineering Task Force

IoE. Internet of Everythings

IoT. Internet of Things

ISO. International Organization for Standardization

JSON. JavaScript Object Notation

LPWAN. Low-Power Wide-Area Network

M2M. Machine to Machine

MNO. Mobile Network Operator

O2. Oxygen

OTT. Over The Top

REST. Representational State Transfer

RFC. Request for Comments

Router ABC. Always Best Connected

SATDR. Group of Distributed Real Time Systems and Applications

STH. Short Time Historic

TEU. Twenty-foot Equivalent Unit

TMS. Traveling Messaging System

WI-SUN. Wireless Smart Ubiquitous Network

5. Bibliography

[1] https://blogthinkbig.com/8-400-millones-de-dispositivos-estaran-conectados-a-internet-a-finales-de-2017
[2] Government of the United States, Official information regarding the Global Positioning System, https://www.gps.gov/
[3] Adafruit Inudstries, Adafruit Ultimate GPS HAT for Raspberry Pi https://www.adafruit.com/product/2324
[4] Omega engineering, ¿Qué es un sensor termopar? https://es.omega.com/prodinfo/termopares.html
[5] K-type thermocouple reference tables: https://es.omega.com/temperature/pdf/Type_K_Thermocouple_Reference_Table.pdf
[6] Sensirion sensors, Digital Humidity Sensor SHT7x, https://www.sensirion.com/en/environmental-sensors/humidity-sensors/pintype-digital-humidity-sensors/
[7] Omega engineering, ¿Qué es un acelerómetro? https://es.omega.com/prodinfo/acelerometro.html
[8] Rs store, Sensor ultrasónico RS Pro, https://es.rs-online.com/web/p/sensores-de-proximidad-ultrasonicos/2370799/
[9] LoRa Alliance, What is the LoRaWAN Specification? https://www.lora-alliance.org/about-lorawan
[10] LoRa Alliance, A technical overview of LoRa and LoRaWAN, November 2015
[11] LoRa Alliance, LoRaWAN 101A Technical Introduction, 2015
[12] Sigfox, Sigfox technology overview https://www.sigfox.com/en/sigfox-iot-technology-overview
[13] Carles Antón-Haro and Mischa Doler, Machine-to-machine (M2M) Communications, 2015
[14] Pilar Andrés-Maldonado y PabloAmeigeras, NarrowBand IoT Data Transmission Procedures for Massive Machine Type Communications¸ Universidad de Granada
[15] Rohde&Schwarz, NarrowBand Internet of Things Whitepaper, https://www.rohde-schwarz.com/es/aplicaciones/internet-de-las-cosas-en-banda-estrecha-white-paper_230854-314242.html
[16] Activage Project, Report on IoT European Platforms, September 2017, https://iot-epi.eu/
[17] Documentación oficial de FIWARE Orion: https://fiware-orion.readthedocs.io
[18] Official documentation of FIWARE STH Comet https://fiware-sth-comet.readthedocs.io
[19] Official documentation of FIWARE Perseo (CEP) http://fiware-iot-stack.readthedocs.io/en/latest/cep/index.html
[20] MongoDB Inc, What is MongoDB? https://www.mongodb.com/what-is-mongodb
[21] University of Berkeley, Cyber-physical Systems https://ptolemy.berkeley.edu/projects/cps/
[22] Docker Inc., What is Docker, https://www.docker.com/what-docker
[23] Docker Inc., What is a container, https://www.docker.com/what-container
[24] Canonical Group, LXD Introduction, https://linuxcontainers.org/lxd/#
[25] Canonical Group, The no-nonsense way to accelerate your business with containers, February 2017
[26] Maersk Line, What is Remote Container Management, https://www.maerskline.com/shipping/remote-container-management
[27] Loginno, Smart container https://loginno.com/
[28] Traxens, TRAXENS Technology, http://www.traxens.com/en/technology
[29] Pointer Telocation LTD, Cellocator, https://www.cellocator.com/products/cellotrack/
[30] Tritón of GlobeArea, http://www.globearea.com/triton
[31] Weightless, http://www.weightless.org/
[32] Weight Sensor of Nanolike https://www.nanolike.com/fill-level-monitoring-solution/
[33] https://www.u-blox.com/en
[34] Blockstack https://bitcoinexchangeguide.com/kadena/
[35] Kadena https://kadena.io/
[36] https://www.plugintoiot.com/sub-1ghz-modules/#CC1310-module
[37] https://www.wi-sun.org/