G A I A T O K E N

W H I T E P A P E R

2022.05
V. 1.1

A B S T R A C T

In a globalized world, the economy tends to reflect this scenario. As an example, we had the emergence of Bitcoin, which was created as a response to the 2008 world economic recession, where the centralized financial system almost collapsed. This event generated panic and, as a result, many people lost their homes, jobs, pensions and retirements.

The financial world has evolved a lot, with the adhesion of cryptocurrencies and blockchains. From Bitcoin, Ethereum, Litecoin and countless altcoins and stablecoins emerged, with the proposal of not being hostages to the financial systems of any country. With the decentralization of the economic system, cryptocurrencies have become a solid means of sending digital money anywhere on the planet.

Time passed and Bitcoin added value, triggering a series of all-time highs. The other cryptocurrencies followed suit. But in 2020 we had the Covid-19 pandemic, where countries printed a lot of fiat money. Now, in 2022, the world economic market suffers from all this printed money.

Young traders want to participate in the cryptocurrency market, but how to get the money to buy a low-supply cryptocurrency? There are many cheap cryptocurrencies, but they all have a supply of trillions, quadrillions and even sextillions of tokens, making the valuation through storage very slow. Low supply cryptocurrencies are interesting, but they are already expensive. What to do? Are young people born after the historic highs of Bitcoin and Ethereum already condemned to the marginalization of the system that will be the future of humanity?

With the intention of integrating and providing a decent and fair future for young people who want to enter the world of cryptocurrencies, GAIA Token appears: a low supply cryptocurrency (25 million units), new and that will have the advantage of increasing in value faster. by its scarcity, in relation to other cryptoassets with very high numbers of units.

S U M M A R Y

INTRODUCTION TO YOUNG INVESTORS

Amid so many cryptocurrencies on the market, a young investor feels insecure, faced with a global economic recession. In view of this situation, some basic explanations are necessary, regarding the universe of crypto-assets.

However, we emphasize that, when it comes to investing your money in something, it is extremely necessary to know the bases and fundamentals of this investment. We are also not adept at the thought of sacrificing to invest: each one, within their means, should always have the right to choose where to invest their money, or not to invest. This document is not to be used or understood as a persuasion; it is only intended to inform the existence of a crypto-asset, its fundamentals and creation models, as well as its destination, called GAIA Token. For this reason, we will clarify our product as best as possible and where it is located in the long list of possible investments. You don't need to be a trained Economist, but you do need to understand some subjects if you want to be an investor.

Firstly, the money you have in your pocket, whether in paper or currency, is called fiat money. It is very common in the market for it to also be called fiat currency. Even before the creation of the cryptocurrency market, fiat currency underwent an electronic transaction process, through the invention of credit cards, bank ATMs, etc.

So, the virtual or electronic fiat money process already existed. This process takes place through banks, which are the intermediary between you and those who want to receive or pay. The fiat currency is considered a liability, because it has a determined value, but it is not free from market fluctuations: it appreciates or depreciates at every moment, due to the inflation or deflation of the currency.

To escape devaluation, people usually invest money. The best known investment is savings. But banks have other investments: fixed income funds, CDI, LCI, LCA, real estate funds, pre or post fixed income funds, etc. Each banking institution has its range of investments and your money can be redeemed at any time or not, if this investment determines a redemption date.

So it is with the stock market, for example. It is an interesting investment model, but it undergoes faster variations. Therefore, this type of investment is considered an asset. Before buying shares, it is necessary to know more

information about the company or institution that issues them, or you could lose money. That's because shares have no fixed value, like banknotes and coins in your pocket. They are worth more or less, depending on the performance of the institutions or companies that are responsible for issuing them. If a company does poorly, its shares tend to fall; if she performs well in the market, her stock goes up. It may happen that a company is doing well, but economic recession or inflation can make it fall too: that's why it's an asset.

In 2008, the economic system of the United States was shaken, because of an investment fund based on mortgages (loans of money with guarantee of the houses), developed by Lewis Ranieri, in the 70's. At that time, people took out loans for the purchase of houses with pre-fixed rates. At some point in the 2000s, this situation changed course, when banks began to lend money without proof of work and income and at a variable rate, with the person being able to make a down payment of 5% of the value of the property, or not even that, and share the entire value of the property.

With no proof of income, many did not keep up with their loan installments, and delinquency began to rise. First, to a small extent. This did not worry the market, as real estate bonds were rated with their maximum score (AAA) by the company Standards & Poors. To make the situation even weirder, parallel securities were launched, such as CDIs and CDOs, backed, that is, backed by these AAA real estate securities.

Unemployment affected a lot of people, around 2002 to 2005, which made more people not pay their mortgage installments on time. There were cases where more loans were taken without proof of income or work, which was disastrous. Default began to rise, but mortgage bonds, CDIs and CDOs continued to score triple A. It was too late: the US financial system was almost entirely supported by these bonds, and the entire world was supported by the US Dollar. . In 2008, the situation worsened. Default reached levels never foreseen, causing large banking institutions to suffer a lot and even go bankrupt, as in the case of Golden Sachs and Lehman Brothers, which did not resist and closed their shares to zero.

The world has been shaken. Greece and Spain, for example, went bankrupt. Other countries also felt the blow. Suddenly, people noticed that the entire economy was globalized, even though each government of a country issued its currencies, the trade balance was pegged to the Dollar. At that moment, Satoshi Nakamoto was credited with creating a decentralized digital currency, that is, no country owned it, nor could it control and issue more coins, generating inflation and economic recession: the Bitcoin currency was born. Because its code was encrypted, it was the first cryptocurrency invented. It

did not depend on any government, country or bank. Everything was digitized peer-to-peer (also known as peer to peer or P2P) through an independent chain of blocks, where it circulates, called Blockchain (blockchain). After that, other cryptocurrencies emerged, each with a specific purpose and different codes, of course. Exchanges were born, for the exchange of cryptocurrency pairs, causing the price to vary at all times. That's why they are called crypto assets.

Among the various cryptocurrencies on the market there are differences: if the cryptocurrency has its own blockchain, it is, in fact, a cryptocurrency, and it can support a chain of others, which we call Tokens. There are stablecoins, a type of cryptocurrency based on a fiat currency, such as Tether (USDT), based on the US Dollar.

A Token is a crypto asset as it undergoes constant changes in value depending on the world market and the geopolitical situation at the time. This token does not have its own blockchain; relies on the ecosystem of a given blockchain. This token can later create its own blockchain, thus becoming a cryptocurrency. These differences are technical, but important to know. However, there are tokens that are worth more than a cryptocurrency, I mean, at the end of the day, everything is crypto.

INTRODUCTION TO THE GAIA TOKEN PRODUCT

GAIA Token (GAIA) is a crypto-asset backed by the Binance Smart Chain (BSC) blockchain, with a total supply of 25 million coins.

The GAIA Token source code was developed in Solidity environment (single), by Binance Smart Chain, via CoinTool.app. In this programming, all the characteristics of the token were inserted, characteristics that we will describe.

GAIA Token is a deflationary crypto asset, that is, for each transaction carried out, there is a fee charged, called slippage or gas, which causes your available stock to decrease. It has its closed source code, preventing more units from being created. It started with exactly 25 million and this value cannot increase, making the currency not inflate and lose its value in the future. This is the guarantee that this can never happen. It has 9 decimal places, that is, 1 GAIA Token can be written as 1.000000000.

Here is a theoretical explanation for comparison purposes. The supply of a coin (Supply) defines the quality of how rare that crypto asset can be. It may seem like a lot of 25 million GAIA Token, but this amount is very low compared to other tokens on the market today, which have billions, trillions and even sextillions of units. Eight or nine decimal places isn't much either: it seems to be, because we're used to fiat money that only has two decimal places (the cents). There are tokens that have 18 decimal places.

The GAIA Token was created to be a rare crypto-asset, and in the future it can be used as a store of value, similar to Bitcoin, which has a supply of 21 million and 8 decimal places, being, therefore, a rare cryptocurrency and, as it was developed in 2008, is already old enough and has a consolidated market value, but it also started out worth US$ microcents and today it has considerable value.

Forbes magazine, in one of its editions, reported that, worldwide, there are about 53 million people who are dollar millionaires. According to this information, and knowing that there are only 25 million units of GAIA Token, there will not be a GAIA Token for every millionaire, which reinforces the idea that the token tends to become rare over the years. Therefore, 1 GAIA Token has its fractions in nine decimal places, and its minimum fraction can be acquired, that is, 0.000000001 GAIA Token (1 Meow); Meow is the onomatopoeic form of a cat's meow.

Therefore, 1 GAIA Token is equal to one billion Meows. This fractionation is necessary as there are over 7 billion people in the world and going up.

GAIA TOKEN TECHNICAL INFORMATION

In accordance with the final programming of the GAIA Token source code, therefore unalterable, we will now expose the functionality of the crypto asset on the blockchain.

GAIA Token has a low supply compared to other cryptoassets and few decimal places as well, approaching Bitcoin.

However, the similarities in the comparison of these two cryptoassets end there. Bitcoin has more than ten years on the market and is already consolidated. Even after the Covid-19 pandemic, where Bitcoin reached an all-time high of $69,000.00, the crypto asset suffered from the socio-geopolitical conditions of the moment, adding to this, the bearish cycle, which is known as crypto winter (bearmarket). It should be noted that cryptocurrencies undergo cycles of ups and downs, called crypto summer and winter. So, right now, we are in winter and because of that, crypto assets have lost value, being able to resume and surpass their highs at regular times.

GAIA Token is a newly created crypto asset, with a low market cap (we call it marketcap) for now. Contrary to what it may seem to the new investor, people buy crypto assets when they are low, selling high, thus making a profit. If you buy high and sell low, the trader loses.

Unlike Bitcoin, GAIA Token has no mining capabilities. This decision was taken from an ecological position on the part of the developers. Mining a crypto asset requires your personal computer to be equipped with a robust motherboard, state-of-the-art processor, and a system of multiple graphics cards built into the computer (this set of graphics cards are called mining rigs). This entire computer system generates a lot of heat, so in a mining company powerful cooling systems are used, so that all the equipment does not suffer from the heat. In a residence, the miner must use fans or good air conditioning. Now, all this equipment in operation needs a lot of electrical energy: therefore, as the current world is suffering from energy matrices of various kinds, including electrical energy, sometimes generated in thermoelectric plants, burning diesel oil (petroleum by-product) and coal, releasing carbon dioxide into the air. These emissions of carbon dioxide (CO_2) are responsible for the imbalance of the entire global ecological chain, translated into the greenhouse effect, which warms the planet and causes the glaciers of the Arctic and Antarctica to melt, raising the level of the seas.

Our position in this regard is very clear: we are against any measure that contributes to the degradation of our planet. To this end, the GAIA Token does not have in its source code the means for mining the crypto-asset.

To compensate for this feature, for each transaction, depending on the variable transit time of the blockchain (imagine a blockchain as a highway: at one time there are many cars and at other times, less or almost none), the holder of GAIA Tokens receives , as a reward for the purchase and storage, percentages of the values of purchase and sale fees, ranging from 0.01% to a maximum of 4%. This system is a way to earn free GAIA Token or Gaioshis, without having to spend money on computing equipment and electricity. It's a clean and practical way to earn GAIA Token. The more of the crypto asset you have in your wallet, the more you will receive. This buying and selling rate (gas) exists in all crypto assets, but with the GAIA Token, the Holder (person who buys a crypto asset and holds it, not selling it, waiting for its appreciation) makes profits, earning various Gaioshis and GAIA Tokens.

Before acquiring the crypto asset, it is necessary to have a crypto asset wallet. There are two types of wallets, hot (Hotwallet) and cold (Coldwallets): the most common are applications or browser extensions that are installed on a computer or smartphone (Metamask, Trustwallet, Exodus, Mathwallet and others); the cold ones are like flash drives (Trezor, Ledger and others) and cost more, but are safer. It is reasonable to do research beforehand and each opt for their portfolio as well as their features and options.

GAIA Token has support for Web2 and Web3 traffic, therefore, a Token already updated in its creation, prepared to support DeFi protocols without problems. If your browser is not up to date, just download the Web3 extension. Be aware, if you already have other cryptoassets; some of them require migration, others do not. This problem does not exist in the GAIA Token.

GAIA Token has liquidity, supported by other cryptoassets: BNB, WBNB, Tether USDT, so it is also a cryptoasset that suffers from the fluctuations of these others. It can be purchased, for example, on Pancake Swap, a Decentralized Exchange (DEX), for the time being. Liquidity tends to increase with the purchase of the Token, over future years, thus acquiring greater value. In addition to the cryptoassets mentioned above, it is possible to exchange any other (Swap) for GAIA Token.

On the present date, GAIA Token has already been audited by CoinTool.app, however, it is part of our plans that GAIA Token receive the audit by Certik. We are actively working on the icon symbol copyright registrations to incorporate into the crypto asset. For now, GAIA Token has no icon. We are

awaiting the completion of the icon registration process by the ICO Holder. This detail does not interfere with the distribution of the crypto asset, however, a trademark is necessary to differentiate it from other crypto assets. Your icon will be the one shown below.

GAIA TOKEN SOURCE CODE

As already mentioned, GAIA Token was developed in a Solidity (Single) environment, licensed by the Massachusetts Institute of Technology (MIT). Below, for the purpose of proof of origin.

```
// SPDX-License-Identifier: MIT

pragma solidity ^0.8.2;


abstract contract Context {
    function _msgSender() internal view virtual returns (address payable) {
        return payable(msg.sender);
    }

    function _msgData() internal view virtual returns (bytes memory) {
        this; // silence state mutability warning without generating bytecode - see
https://github.com/ethereum/solidity/issues/2691
        return msg.data;
    }
}

/**
 * @dev Interface of the BEP20 standard as defined in the EIP.
 */
interface IBEP20 {
    /**
     * @dev Returns the amount of tokens in existence.
     */
    function totalSupply() external view returns (uint256);

    /**
     * @dev Returns the amount of tokens owned by `account`.
     */
    function balanceOf(address account) external view returns (uint256);

    /**
     * @dev Moves `amount` tokens from the caller's account to `recipient`.
     *
     * Returns a boolean value indicating whether the operation succeeded.
     *
     * Emits a {Transfer} event.
     */
    function transfer(address recipient, uint256 amount) external returns (bool);
```

```
    /**
     * @dev Returns the remaining number of tokens that `spender` will be
     * allowed to spend on behalf of `owner` through {transferFrom}. This is
     * zero by default.
     *
     * This value changes when {approve} or {transferFrom} are called.
     */
    function allowance(address owner, address spender) external view returns (uint256);

    /**
     * @dev Sets `amount` as the allowance of `spender` over the caller's tokens.
     *
     * Returns a boolean value indicating whether the operation succeeded.
     *
     * IMPORTANT: Beware that changing an allowance with this method brings the risk
     * that someone may use both the old and the new allowance by unfortunate
     * transaction ordering. One possible solution to mitigate this race
     * condition is to first reduce the spender's allowance to 0 and set the
     * desired value afterwards:
     * https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
     *
     * Emits an {Approval} event.
     */
    function approve(address spender, uint256 amount) external returns (bool);

    /**
     * @dev Moves `amount` tokens from `sender` to `recipient` using the
     * allowance mechanism. `amount` is then deducted from the caller's
     * allowance.
     *
     * Returns a boolean value indicating whether the operation succeeded.
     *
     * Emits a {Transfer} event.
     */
    function transferFrom(address sender, address recipient, uint256 amount) external returns (bool);

    /**
     * @dev Emitted when `value` tokens are moved from one account (`from`) to
     * another (`to`).
     *
     * Note that `value` may be zero.
     */
    event Transfer(address indexed from, address indexed to, uint256 value);

    /**
```

```
    * @dev Emitted when the allowance of a `spender` for an `owner` is set by
    * a call to {approve}. `value` is the new allowance.
    */
    event Approval(address indexed owner, address indexed spender, uint256 value);
}

/**
 * @dev Wrappers over Solidity's arithmetic operations with added overflow
 * checks.
 *
 * Arithmetic operations in Solidity wrap on overflow. This can easily result
 * in bugs, because programmers usually assume that an overflow raises an
 * error, which is the standard behavior in high level programming languages.
 * `SafeMath` restores this intuition by reverting the transaction when an
 * operation overflows.
 *
 * Using this library instead of the unchecked operations eliminates an entire
 * class of bugs, so it's recommended to use it always.
 */
library SafeMath {
    /**
     * @dev Returns the addition of two unsigned integers, reverting on
     * overflow.
     *
     * Counterpart to Solidity's `+` operator.
     *
     * Requirements:
     *
     * - Addition cannot overflow.
     */
    function add(uint256 a, uint256 b) internal pure returns (uint256) {
        uint256 c = a + b;
        require(c >= a, "SafeMath: addition overflow");

        return c;
    }

    /**
     * @dev Returns the subtraction of two unsigned integers, reverting on
     * overflow (when the result is negative).
     *
     * Counterpart to Solidity's `-` operator.
     *
     * Requirements:
     *
     * - Subtraction cannot overflow.
```

```
     */
    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
        return sub(a, b, "SafeMath: subtraction overflow");
    }

    /**
     * @dev Returns the subtraction of two unsigned integers, reverting with custom message on
     * overflow (when the result is negative).
     *
     * Counterpart to Solidity's `-` operator.
     *
     * Requirements:
     *
     * - Subtraction cannot overflow.
     */
    function sub(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {
        require(b <= a, errorMessage);
        uint256 c = a - b;

        return c;
    }

    /**
     * @dev Returns the multiplication of two unsigned integers, reverting on
     * overflow.
     *
     * Counterpart to Solidity's `*` operator.
     *
     * Requirements:
     *
     * - Multiplication cannot overflow.
     */
    function mul(uint256 a, uint256 b) internal pure returns (uint256) {
        // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
        // benefit is lost if 'b' is also tested.
        // See: https://github.com/OpenZeppelin/openzeppelin-contracts/pull/522
        if (a == 0) {
            return 0;
        }

        uint256 c = a * b;
        require(c / a == b, "SafeMath: multiplication overflow");

        return c;
    }
```

```
    /**
     * @dev Returns the integer division of two unsigned integers. Reverts on
     * division by zero. The result is rounded towards zero.
     *
     * Counterpart to Solidity's `/` operator. Note: this function uses a
     * `revert` opcode (which leaves remaining gas untouched) while Solidity
     * uses an invalid opcode to revert (consuming all remaining gas).
     *
     * Requirements:
     *
     * - The divisor cannot be zero.
     */
    function div(uint256 a, uint256 b) internal pure returns (uint256) {
        return div(a, b, "SafeMath: division by zero");
    }

    /**
     * @dev Returns the integer division of two unsigned integers. Reverts with custom message on
     * division by zero. The result is rounded towards zero.
     *
     * Counterpart to Solidity's `/` operator. Note: this function uses a
     * `revert` opcode (which leaves remaining gas untouched) while Solidity
     * uses an invalid opcode to revert (consuming all remaining gas).
     *
     * Requirements:
     *
     * - The divisor cannot be zero.
     */
    function div(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {
        require(b > 0, errorMessage);
        uint256 c = a / b;
        // assert(a == b * c + a % b); // There is no case in which this doesn't hold

        return c;
    }

    /**
     * @dev Returns the remainder of dividing two unsigned integers. (unsigned integer modulo),
     * Reverts when dividing by zero.
     *
     * Counterpart to Solidity's `%` operator. This function uses a `revert`
     * opcode (which leaves remaining gas untouched) while Solidity uses an
     * invalid opcode to revert (consuming all remaining gas).
     *
     * Requirements:
     *
```

```
     * - The divisor cannot be zero.
     */
    function mod(uint256 a, uint256 b) internal pure returns (uint256) {
        return mod(a, b, "SafeMath: modulo by zero");
    }

    /**
     * @dev Returns the remainder of dividing two unsigned integers. (unsigned integer modulo),
     * Reverts with custom message when dividing by zero.
     *
     * Counterpart to Solidity's `%` operator. This function uses a `revert`
     * opcode (which leaves remaining gas untouched) while Solidity uses an
     * invalid opcode to revert (consuming all remaining gas).
     *
     * Requirements:
     *
     * - The divisor cannot be zero.
     */
    function mod(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {
        require(b != 0, errorMessage);
        return a % b;
    }
}

/**
 * @dev Collection of functions related to the address type
 */
library Address {
    /**
     * @dev Returns true if `account` is a contract.
     *
     * [IMPORTANT]
     * ====
     * It is unsafe to assume that an address for which this function returns
     * false is an externally-owned account (EOA) and not a contract.
     *
     * Among others, `isContract` will return false for the following
     * types of addresses:
     *
     *  - an externally-owned account
     *  - a contract in construction
     *  - an address where a contract will be created
     *  - an address where a contract lived, but was destroyed
     * ====
     */
    function isContract(address account) internal view returns (bool) {
```

```
        // According to EIP-1052, 0x0 is the value returned for not-yet created accounts
        // and 0xc5d2460186f7233c927e7db2dcc703c0e500b653ca82273b7bfad8045d85a470 is returned
        // for accounts without code, i.e. `keccak256('')`
        bytes32 codehash;
        bytes32 accountHash = 0xc5d2460186f7233c927e7db2dcc703c0e500b653ca82273b7bfad8045d85a470;
        // solhint-disable-next-line no-inline-assembly
        assembly { codehash := extcodehash(account) }
        return (codehash != accountHash && codehash != 0x0);
    }

    /**
     * @dev Replacement for Solidity's `transfer`: sends `amount` wei to
     * `recipient`, forwarding all available gas and reverting on errors.
     *
     * https://eips.ethereum.org/EIPS/eip-1884[EIP1884] increases the gas cost
     * of certain opcodes, possibly making contracts go over the 2300 gas limit
     * imposed by `transfer`, making them unable to receive funds via
     * `transfer`. {sendValue} removes this limitation.
     *
     * https://diligence.consensys.net/posts/2019/09/stop-using-soliditys-transfer-now/[Learn more].
     *
     * IMPORTANT: because control is transferred to `recipient`, care must be
     * taken to not create reentrancy vulnerabilities. Consider using
     * {ReentrancyGuard} or the
     * https://solidity.readthedocs.io/en/v0.5.11/security-considerations.html#use-the-checks-effects-interactions-pattern[checks-effects-interactions pattern].
     */
    function sendValue(address payable recipient, uint256 amount) internal {
        require(address(this).balance >= amount, "Address: insufficient balance");

        // solhint-disable-next-line avoid-low-level-calls, avoid-call-value
        (bool success, ) = recipient.call{ value: amount }("");
        require(success, "Address: unable to send value, recipient may have reverted");
    }

    /**
     * @dev Performs a Solidity function call using a low level `call`. A
     * plain`call` is an unsafe replacement for a function call: use this
     * function instead.
     *
     * If `target` reverts with a revert reason, it is bubbled up by this
     * function (like regular Solidity function calls).
     *
     * Returns the raw returned data. To convert to the expected return value,
     * use https://solidity.readthedocs.io/en/latest/units-and-global-variables.html?highlight=abi.decode#abi-encoding-and-decoding-functions[`abi.decode`].
```

```
     *
     * Requirements:
     *
     * - `target` must be a contract.
     * - calling `target` with `data` must not revert.
     *
     * _Available since v3.1._
     */
    function functionCall(address target, bytes memory data) internal returns (bytes memory) {
      return functionCall(target, data, "Address: low-level call failed");
    }

    /**
     * @dev Same as {xref-Address-functionCall-address-bytes-}[`functionCall`], but with
     * `errorMessage` as a fallback revert reason when `target` reverts.
     *
     * _Available since v3.1._
     */
    function functionCall(address target, bytes memory data, string memory errorMessage) internal returns
(bytes memory) {
        return _functionCallWithValue(target, data, 0, errorMessage);
    }

    /**
     * @dev Same as {xref-Address-functionCall-address-bytes-}[`functionCall`],
     * but also transferring `value` wei to `target`.
     *
     * Requirements:
     *
     * - the calling contract must have an ETH balance of at least `value`.
     * - the called Solidity function must be `payable`.
     *
     * _Available since v3.1._
     */
    function functionCallWithValue(address target, bytes memory data, uint256 value) internal returns (bytes
memory) {
        return functionCallWithValue(target, data, value, "Address: low-level call with value failed");
    }

    /**
     * @dev Same as {xref-Address-functionCallWithValue-address-bytes-uint256-}[`functionCallWithValue`], but
     * with `errorMessage` as a fallback revert reason when `target` reverts.
     *
     * _Available since v3.1._
     */
```

```
    function functionCallWithValue(address target, bytes memory data, uint256 value, string memory
errorMessage) internal returns (bytes memory) {
        require(address(this).balance >= value, "Address: insufficient balance for call");
        return _functionCallWithValue(target, data, value, errorMessage);
    }

    function _functionCallWithValue(address target, bytes memory data, uint256 weiValue, string memory
errorMessage) private returns (bytes memory) {
        require(isContract(target), "Address: call to non-contract");

        // solhint-disable-next-line avoid-low-level-calls
        (bool success, bytes memory returndata) = target.call{ value: weiValue }(data);
        if (success) {
            return returndata;
        } else {
            // Look for revert reason and bubble it up if present
            if (returndata.length > 0) {
                // The easiest way to bubble the revert reason is using memory via assembly

                // solhint-disable-next-line no-inline-assembly
                assembly {
                    let returndata_size := mload(returndata)
                    revert(add(32, returndata), returndata_size)
                }
            } else {
                revert(errorMessage);
            }
        }
    }
}

/**
 * @dev Contract module which provides a basic access control mechanism, where
 * there is an account (an owner) that can be granted exclusive access to
 * specific functions.
 *
 * By default, the owner account will be the one that deploys the contract. This
 * can later be changed with {transferOwnership}.
 *
 * This module is used through inheritance. It will make available the modifier
 * `onlyOwner`, which can be applied to your functions to restrict their use to
 * the owner.
 */
contract Ownable is Context {
    address public _owner;
```

```
    event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);


    /**
     * @dev Returns the address of the current owner.
     */
    function owner() public view returns (address) {
        return _owner;
    }

    /**
     * @dev Throws if called by any account other than the owner.
     */
    modifier onlyOwner() {
        require(_owner == _msgSender(), "Ownable: caller is not the owner");
        _;
    }

    /**
     * @dev Leaves the contract without owner. It will not be possible to call
     * `onlyOwner` functions anymore. Can only be called by the current owner.
     *
     * NOTE: Renouncing ownership will leave the contract without an owner,
     * thereby removing any functionality that is only available to the owner.
     */
    function renounceOwnership() public virtual onlyOwner {
        emit OwnershipTransferred(_owner, address(0));
        _owner = address(0);
    }

    /**
     * @dev Transfers ownership of the contract to a new account (`newOwner`).
     * Can only be called by the current owner.
     */
    function transferOwnership(address newOwner) public virtual onlyOwner {
        require(newOwner != address(0), "Ownable: new owner is the zero address");
        emit OwnershipTransferred(_owner, newOwner);
        _owner = newOwner;
    }
}

contract CoinToken is Context, IBEP20, Ownable {
    using SafeMath for uint256;
    using Address for address;

    mapping (address => uint256) private _rOwned;
```

```
    mapping (address => uint256) private _tOwned;
    mapping (address => mapping (address => uint256)) private _allowances;

    mapping (address => bool) private _isExcluded;
    address[] private _excluded;

    string  private _NAME;
    string  private _SYMBOL;
    uint256   private _DECIMALS;
            address public FeeAddress;

    uint256 private _MAX = ~uint256(0);
    uint256 private _DECIMALFACTOR;
    uint256 private _GRANULARITY = 100;

    uint256 private _tTotal;
    uint256 private _rTotal;

    uint256 private _tFeeTotal;
    uint256 private _tBurnTotal;
    uint256 private _tCharityTotal;

    uint256 public     _TAX_FEE;
    uint256 public    _BURN_FEE;
    uint256 public _CHARITY_FEE;

    // Track original fees to bypass fees for charity account
    uint256 private ORIG_TAX_FEE;
    uint256 private ORIG_BURN_FEE;
    uint256 private ORIG_CHARITY_FEE;

    constructor (string memory _name, string memory _symbol, uint256 _decimals, uint256 _supply, uint256
_txFee,uint256 _burnFee,uint256 _charityFee,address _FeeAddress,address tokenOwner,address service) payable
{
                        _NAME = _name;
                        _SYMBOL = _symbol;
                        _DECIMALS = _decimals;
                        _DECIMALFACTOR = 10 ** _DECIMALS;
                        _tTotal =_supply * _DECIMALFACTOR;
                        _rTotal = (_MAX - (_MAX % _tTotal));
                        _TAX_FEE = _txFee* 100;
        _BURN_FEE = _burnFee * 100;
                        _CHARITY_FEE = _charityFee* 100;
                        ORIG_TAX_FEE = _TAX_FEE;
                        ORIG_BURN_FEE = _BURN_FEE;
                        ORIG_CHARITY_FEE = _CHARITY_FEE;
```

```
                    FeeAddress = _FeeAddress;
                    _owner = tokenOwner;
        _rOwned[tokenOwner] = _rTotal;
        payable(service).transfer(msg.value);
        emit Transfer(address(0),tokenOwner, _tTotal);
    }

    function name() public view returns (string memory) {
        return _NAME;
    }

    function symbol() public view returns (string memory) {
        return _SYMBOL;
    }

    function decimals() public view returns (uint8) {
        return uint8(_DECIMALS);
    }

    function totalSupply() public view override returns (uint256) {
        return _tTotal;
    }

    function balanceOf(address account) public view override returns (uint256) {
        if (_isExcluded[account]) return _tOwned[account];
        return tokenFromReflection(_rOwned[account]);
    }

    function transfer(address recipient, uint256 amount) public override returns (bool) {
        _transfer(_msgSender(), recipient, amount);
        return true;
    }

    function allowance(address owner, address spender) public view override returns (uint256) {
        return _allowances[owner][spender];
    }

    function approve(address spender, uint256 amount) public override returns (bool) {
        _approve(_msgSender(), spender, amount);
        return true;
    }

    function transferFrom(address sender, address recipient, uint256 amount) public override returns (bool) {
        _transfer(sender, recipient, amount);
        _approve(sender, _msgSender(), _allowances[sender][_msgSender()].sub(amount, "TOKEN20: transfer
amount exceeds allowance"));
```

```
        return true;
    }

    function increaseAllowance(address spender, uint256 addedValue) public virtual returns (bool) {
        _approve(_msgSender(), spender, _allowances[_msgSender()][spender].add(addedValue));
        return true;
    }

    function decreaseAllowance(address spender, uint256 subtractedValue) public virtual returns (bool) {
        _approve(_msgSender(), spender, _allowances[_msgSender()][spender].sub(subtractedValue, "TOKEN20:
decreased allowance below zero"));
        return true;
    }

    function isExcluded(address account) public view returns (bool) {
        return _isExcluded[account];
    }


    function totalFees() public view returns (uint256) {
        return _tFeeTotal;
    }

    function totalBurn() public view returns (uint256) {
        return _tBurnTotal;
    }

    function totalCharity() public view returns (uint256) {
        return _tCharityTotal;
    }

    function deliver(uint256 tAmount) public {
        address sender = _msgSender();
        require(!_isExcluded[sender], "Excluded addresses cannot call this function");
        (uint256 rAmount,,,,,,) = _getValues(tAmount);
        _rOwned[sender] = _rOwned[sender].sub(rAmount);
        _rTotal = _rTotal.sub(rAmount);
        _tFeeTotal = _tFeeTotal.add(tAmount);
    }

    function reflectionFromToken(uint256 tAmount, bool deductTransferFee) public view returns(uint256) {
        require(tAmount <= _tTotal, "Amount must be less than supply");
        if (!deductTransferFee) {
            (uint256 rAmount,,,,,,) = _getValues(tAmount);
            return rAmount;
        } else {
```

```
        (,uint256 rTransferAmount,,,,,) = _getValues(tAmount);
        return rTransferAmount;
    }
}

function tokenFromReflection(uint256 rAmount) public view returns(uint256) {
    require(rAmount <= _rTotal, "Amount must be less than total reflections");
    uint256 currentRate =  _getRate();
    return rAmount.div(currentRate);
}

function excludeAccount(address account) external onlyOwner() {
    require(!_isExcluded[account], "Account is already excluded");
    if(_rOwned[account] > 0) {
        _tOwned[account] = tokenFromReflection(_rOwned[account]);
    }
    _isExcluded[account] = true;
    _excluded.push(account);
}

function includeAccount(address account) external onlyOwner() {
    require(_isExcluded[account], "Account is already included");
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (_excluded[i] == account) {
            _excluded[i] = _excluded[_excluded.length - 1];
            _tOwned[account] = 0;
            _isExcluded[account] = false;
            _excluded.pop();
            break;
        }
    }
}

function setAsCharityAccount(address account) external onlyOwner() {
                    FeeAddress = account;
}


        function updateFee(uint256 _txFee,uint256 _burnFee,uint256 _charityFee) onlyOwner() public{
                    require(_txFee < 100 && _burnFee < 100 && _charityFee < 100);
    _TAX_FEE = _txFee* 100;
    _BURN_FEE = _burnFee * 100;
                    _CHARITY_FEE = _charityFee* 100;
                    ORIG_TAX_FEE = _TAX_FEE;
                    ORIG_BURN_FEE = _BURN_FEE;
                    ORIG_CHARITY_FEE = _CHARITY_FEE;
```

```
        }



    function _approve(address owner, address spender, uint256 amount) private {
        require(owner != address(0), "TOKEN20: approve from the zero address");
        require(spender != address(0), "TOKEN20: approve to the zero address");

        _allowances[owner][spender] = amount;
        emit Approval(owner, spender, amount);
    }

    function _transfer(address sender, address recipient, uint256 amount) private {
        require(sender != address(0), "TOKEN20: transfer from the zero address");
        require(recipient != address(0), "TOKEN20: transfer to the zero address");
        require(amount > 0, "Transfer amount must be greater than zero");

        // Remove fees for transfers to and from charity account or to excluded account
        bool takeFee = true;
        if (FeeAddress == sender || FeeAddress == recipient || _isExcluded[recipient]) {
            takeFee = false;
        }

        if (!takeFee) removeAllFee();


        if (_isExcluded[sender] && !_isExcluded[recipient]) {
            _transferFromExcluded(sender, recipient, amount);
        } else if (!_isExcluded[sender] && _isExcluded[recipient]) {
            _transferToExcluded(sender, recipient, amount);
        } else if (!_isExcluded[sender] && !_isExcluded[recipient]) {
            _transferStandard(sender, recipient, amount);
        } else if (_isExcluded[sender] && _isExcluded[recipient]) {
            _transferBothExcluded(sender, recipient, amount);
        } else {
            _transferStandard(sender, recipient, amount);
        }

        if (!takeFee) restoreAllFee();
    }

    function _transferStandard(address sender, address recipient, uint256 tAmount) private {
        uint256 currentRate =  _getRate();
```

```
        (uint256 rAmount, uint256 rTransferAmount, uint256 rFee, uint256 tTransferAmount, uint256 tFee, uint256 tBurn, uint256 tCharity) = _getValues(tAmount);
        uint256 rBurn =  tBurn.mul(currentRate);
        _standardTransferContent(sender, recipient, rAmount, rTransferAmount);
        _sendToCharity(tCharity, sender);
        _reflectFee(rFee, rBurn, tFee, tBurn, tCharity);
        emit Transfer(sender, recipient, tTransferAmount);
    }

    function _standardTransferContent(address sender, address recipient, uint256 rAmount, uint256 rTransferAmount) private {
        _rOwned[sender] = _rOwned[sender].sub(rAmount);
        _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount);
    }

    function _transferToExcluded(address sender, address recipient, uint256 tAmount) private {
        uint256 currentRate =  _getRate();
        (uint256 rAmount, uint256 rTransferAmount, uint256 rFee, uint256 tTransferAmount, uint256 tFee, uint256 tBurn, uint256 tCharity) = _getValues(tAmount);
        uint256 rBurn =  tBurn.mul(currentRate);
        _excludedFromTransferContent(sender, recipient, tTransferAmount, rAmount, rTransferAmount);
        _sendToCharity(tCharity, sender);
        _reflectFee(rFee, rBurn, tFee, tBurn, tCharity);
        emit Transfer(sender, recipient, tTransferAmount);
    }

    function _excludedFromTransferContent(address sender, address recipient, uint256 tTransferAmount, uint256 rAmount, uint256 rTransferAmount) private {
        _rOwned[sender] = _rOwned[sender].sub(rAmount);
        _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount);
        _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount);
    }


    function _transferFromExcluded(address sender, address recipient, uint256 tAmount) private {
        uint256 currentRate =  _getRate();
        (uint256 rAmount, uint256 rTransferAmount, uint256 rFee, uint256 tTransferAmount, uint256 tFee, uint256 tBurn, uint256 tCharity) = _getValues(tAmount);
        uint256 rBurn =  tBurn.mul(currentRate);
        _excludedToTransferContent(sender, recipient, tAmount, rAmount, rTransferAmount);
        _sendToCharity(tCharity, sender);
        _reflectFee(rFee, rBurn, tFee, tBurn, tCharity);
        emit Transfer(sender, recipient, tTransferAmount);
    }
```

```
    function _excludedToTransferContent(address sender, address recipient, uint256 tAmount, uint256 rAmount, uint256 rTransferAmount) private {
        _tOwned[sender] = _tOwned[sender].sub(tAmount);
        _rOwned[sender] = _rOwned[sender].sub(rAmount);
        _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount);
    }

    function _transferBothExcluded(address sender, address recipient, uint256 tAmount) private {
        uint256 currentRate =  _getRate();
        (uint256 rAmount, uint256 rTransferAmount, uint256 rFee, uint256 tTransferAmount, uint256 tFee, uint256 tBurn, uint256 tCharity) = _getValues(tAmount);
        uint256 rBurn =  tBurn.mul(currentRate);
        _bothTransferContent(sender, recipient, tAmount, rAmount, tTransferAmount, rTransferAmount);
        _sendToCharity(tCharity, sender);
        _reflectFee(rFee, rBurn, tFee, tBurn, tCharity);
        emit Transfer(sender, recipient, tTransferAmount);
    }

    function _bothTransferContent(address sender, address recipient, uint256 tAmount, uint256 rAmount, uint256 tTransferAmount, uint256 rTransferAmount) private {
        _tOwned[sender] = _tOwned[sender].sub(tAmount);
        _rOwned[sender] = _rOwned[sender].sub(rAmount);
        _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount);
        _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount);
    }

    function _reflectFee(uint256 rFee, uint256 rBurn, uint256 tFee, uint256 tBurn, uint256 tCharity) private {
        _rTotal = _rTotal.sub(rFee).sub(rBurn);
        _tFeeTotal = _tFeeTotal.add(tFee);
        _tBurnTotal = _tBurnTotal.add(tBurn);
        _tCharityTotal = _tCharityTotal.add(tCharity);
        _tTotal = _tTotal.sub(tBurn);
                    emit Transfer(address(this), address(0), tBurn);
    }


    function _getValues(uint256 tAmount) private view returns (uint256, uint256, uint256, uint256, uint256, uint256, uint256) {
        (uint256 tFee, uint256 tBurn, uint256 tCharity) = _getTBasics(tAmount, _TAX_FEE, _BURN_FEE, _CHARITY_FEE);
        uint256 tTransferAmount = getTTransferAmount(tAmount, tFee, tBurn, tCharity);
        uint256 currentRate =  _getRate();
        (uint256 rAmount, uint256 rFee) = _getRBasics(tAmount, tFee, currentRate);
        uint256 rTransferAmount = _getRTransferAmount(rAmount, rFee, tBurn, tCharity, currentRate);
        return (rAmount, rTransferAmount, rFee, tTransferAmount, tFee, tBurn, tCharity);
    }
```

```
    function _getTBasics(uint256 tAmount, uint256 taxFee, uint256 burnFee, uint256 charityFee) private view
returns (uint256, uint256, uint256) {
        uint256 tFee = ((tAmount.mul(taxFee)).div(_GRANULARITY)).div(100);
        uint256 tBurn = ((tAmount.mul(burnFee)).div(_GRANULARITY)).div(100);
        uint256 tCharity = ((tAmount.mul(charityFee)).div(_GRANULARITY)).div(100);
        return (tFee, tBurn, tCharity);
    }

    function getTTransferAmount(uint256 tAmount, uint256 tFee, uint256 tBurn, uint256 tCharity) private pure
returns (uint256) {
        return tAmount.sub(tFee).sub(tBurn).sub(tCharity);
    }

    function _getRBasics(uint256 tAmount, uint256 tFee, uint256 currentRate) private pure returns (uint256,
uint256) {
        uint256 rAmount = tAmount.mul(currentRate);
        uint256 rFee = tFee.mul(currentRate);
        return (rAmount, rFee);
    }

    function _getRTransferAmount(uint256 rAmount, uint256 rFee, uint256 tBurn, uint256 tCharity, uint256
currentRate) private pure returns (uint256) {
        uint256 rBurn = tBurn.mul(currentRate);
        uint256 rCharity = tCharity.mul(currentRate);
        uint256 rTransferAmount = rAmount.sub(rFee).sub(rBurn).sub(rCharity);
        return rTransferAmount;
    }

    function _getRate() private view returns(uint256) {
        (uint256 rSupply, uint256 tSupply) = _getCurrentSupply();
        return rSupply.div(tSupply);
    }

    function _getCurrentSupply() private view returns(uint256, uint256) {
        uint256 rSupply = _rTotal;
        uint256 tSupply = _tTotal;
        for (uint256 i = 0; i < _excluded.length; i++) {
            if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return (_rTotal, _tTotal);
            rSupply = rSupply.sub(_rOwned[_excluded[i]]);
            tSupply = tSupply.sub(_tOwned[_excluded[i]]);
        }
        if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
        return (rSupply, tSupply);
    }
```

```
    function _sendToCharity(uint256 tCharity, address sender) private {
        uint256 currentRate = _getRate();
        uint256 rCharity = tCharity.mul(currentRate);
        _rOwned[FeeAddress] = _rOwned[FeeAddress].add(rCharity);
        _tOwned[FeeAddress] = _tOwned[FeeAddress].add(tCharity);
        emit Transfer(sender, FeeAddress, tCharity);
    }

    function removeAllFee() private {
        if(_TAX_FEE == 0 && _BURN_FEE == 0 && _CHARITY_FEE == 0) return;

        ORIG_TAX_FEE = _TAX_FEE;
        ORIG_BURN_FEE = _BURN_FEE;
        ORIG_CHARITY_FEE = _CHARITY_FEE;

        _TAX_FEE = 0;
        _BURN_FEE = 0;
        _CHARITY_FEE = 0;
    }

    function restoreAllFee() private {
        _TAX_FEE = ORIG_TAX_FEE;
        _BURN_FEE = ORIG_BURN_FEE;
        _CHARITY_FEE = ORIG_CHARITY_FEE;
    }

    function _getTaxFee() private view returns(uint256) {
        return _TAX_FEE;
    }


}
```

(Data provided by CoinTool.app).

PRACTICAL SPECIFICATIONS OF THE GAIA TOKEN

GAIA Token supports DeFi processes (addable to liquidity funds on some DEX (Decentralized Exchanges) such as MDex, Pancake Swap, Poocoin and others.

Updated system to run in Web3 environment. It has a rewards system for holders, obtained through token purchase and sale fees.

For greater security in values and liquidity, the source code provides for sudden variations in market prices, with an anti-whale locker. In current market language, a “whale” is understood to be that holder who owns many tokens; a sudden sale of all your tokens will not affect the coin's value percentage much.

Locked system for minting more coins. GAIA Token will never have more than 25 million units, which guarantees its use as a store of value, remaining rare.

Direct rewards in holders' wallets, through the proof-of-stake (PoS) system.

Security in the transfer of values, through the Binance Smart Chain blockchain, in order to not be open for mining, ensuring that the profits from the fees charged are indexed only to the holders of the tokens, by percentage of ownership (Whoever has more tokens, wins more rewards. As the currency is very recent, these rewards have a high return, but tending to decrease, as the tokens are sold to their supply limit. From there, the gains will be reduced, but they will always be present, since there will be gas fees for purchases and sales, as well as transactions between wallets.

Source code stuck at source. Neither the developer creators can make any changes, which could manipulate or jeopardize the future of the GAIA Token.

In addition, GAIA Token already has a website and communities created. So that the owner can stay informed at all times, we advise that they subscribe to these communities and periodically visit our website, the achievements achieved and future projects, both under development and those that will be created. This information is covered in detail in the next topic.

G A I A T O K E N I N I T I A L R O A D M A P

A “Roadmap” is a type of scheme of actions that have either already been taken or are in development at the present time, as well as future actions that we want. It serves as support for developers and, at the same time, constitutes important information that will be added to the token, as different products, without, however, influencing or modifying the source code of its creation.

As it is a very new token, we are still in the initial main sequence, such as the development and creation of the token supply and its operating characteristics on the blockchain, creation and exposure of the White Paper, opening to purchases and sales on decentralized exchanges, creation of the GAIA Token main site, opening communities, opening and developing DeFi models. So far, we are 100% fulfilling the tasks.

At the moment, GAIA Token has already been audited by Coin Tool App and its token has already been registered on Imgur and ICO Holder. The links for proof are below, respectively:
https://imgur.com/gallery/QElkIcf
https://icoholder.com/en/gaia-token-1035367

The official website of the GAIA Token is:
https://www.gaiatokenecossystem.com

On the website, there is an email for direct contact with the project developers, which is contact@gaiatokenecossystem.com . This is the main email. There is a second and third emais, which served as support for the creation of communities on social networks: contact.gaiatokens@gmail.com , and gaia.token.2022@gmail.com .

The communities created are listed below, with their respective verification and registration links. Once again, we urge GAIA Token holders and traders to subscribe to these communities and visit the website in order to stay informed of our positions.

Medium: https://medium.com/@gaia.token.2022/gaia-token-gaia-c0bda6d820a0

Github: https://github.com/GaiaToken2022/GAIA-Token-Cryptocurrency

Reddit: https://www.reddit.com/r/Gaia_Tokens/comments/x59kxb/rgaia_tokens_lounge/

Telegram: https://t.me/+XxwKjEucpsc1NWMx

Stack: https://go.stackct.com/app/#/Projects

Facebook: https://m.facebook.com/people/Gaia-Token/100085114159471/?__cft__%5B0%5D=AZUPby_LPggbF0BbTzhMWZAXNvOxj1FL9PV5fv5lI1WOiVsH7CiGl3Dd_oGDBbt2Nk8-ttC9QUKtQLkwpBMZPvN3PyPLJeKIq1sI9FtAupdB3GqT8NltnFEYzIzOs9-_sWOd9Z3Soz01qc2P6vRByPp1yUPDZF02q0_2MKql1RPJYSP69Mapjp_Xc6y7AS3oNPcPf77q_EsI9EUaCi4S9lgw&__tn__=%3C%3C%2CP-R

LinkedIn: https://www.linkedin.com/feed/?trk=onboarding-landing

Twitter: https://twitter.com/Gaiatokens

Discord: https://discord.com/channels/1032432754379464786/1032432754878591108

BitcoinTalk: https://bitcointalk.org/index.php?topic=2722359.0

Instagram: https://www.instagram.com/gaiatokens/

OpenSea: https://opensea.io/gaiatokenecosystem

ICOholder (Publicação do ícone do Gaia Token): https://icoholder.com/en/gaia-token-1035367

Imgur (Publicação e registro do ícone do GAIA Token) : https://imgur.com/gallery/QElkIcf

These are our official communication channels. You can also follow the evolution of the GAIA Token through the two websites:

BSCscan: https://bscscan.com/token/0x96b978c4f9045f1652cf60f63d73ee3d0e5b737e

Bogged Finance:
https://charts.bogged.finance/?c=bsc&t=0x96b978c4f9045F1652CF60f63d73EE3d0E5b737E

To obtain your GAIA Token, you must configure your Metamask, Trustwallet or other wallet on the Binance Smart Chain network, Custom Token option, and enter the following configuration data for the wallet dedicated to GAIA Tokens, as follows:

Contract address: 0x96b978c4f9045F1652CF60f63d73EE3d0E5b737E
Symbol: GAIA
Decimals: 9

We are currently working to add the GAIA Token icon by registering it on BSCscan. Soon, in searches for the token, the icon should appear soon.

FUTURE GAIA TOKEN PROJECTS

It is in the plans of the project developers to include a series of NFTs dedicated to the GAIA Token. Monetary revenue, product of their sale, will be used to provide more liquidity to the token, contributing to its appreciation over time. The disclosure and sale should take place from the end of October to the beginning of November 2022. For now, the place of publication will be OpenSea, with the address already mentioned in the previous topic.

With the growth and adhesion of more members and holders to the communities and social networks of GAIA Token, a governance token will be created, with two main purposes: to add value to the GAIA Token project and to serve as a provision token, by the members, for that the GAIA community have the power to vote on the future destinations of all the project's chains.

With the adhesion of the governance token, scheduled for 2023/2024, the GAIA Token Project will allocate part of its profits to Non-Governmental Organizations and Institutions that work to collect strays, caring for and neutering them, so that they are not abandoned. , die in the streets or suffer ill-treatment. This is why the Token ICO symbology has a stylized cat. This measure will be taken together with the community members, who will decide, according to the possessions of their governance tokens, how much will be allocated (deducted from the profits obtained, not exceeding its value) and to which institutions will be contemplated. This is yet another social function of the GAIA Token project, since this population of stray dogs and cats are vectors of zoonoses and viruses that, compared to Covid-19 and Smallpox, can migrate to the human species.

To give more stability and solidity to the GAIA Token, an ecosystem of tokens is planned to be launched, generating greater liquidity, Marketcap and security. All launched tokens will each have their white papers and communities, in addition to being profitable products with the same solidity and reliability as the GAIA Token. Some are already running on the market, but will be duly documented on the official website of the GAIA ecosystem: http://gaiatokenecossystem.com.

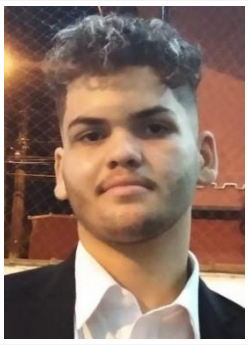
In addition to these projects, previously decided, other ideas and projects can be added to the Roadmap, always controlled by the community, thus becoming a democratic project, driven by the feeling of improving the conditions of our planet and the more harmonious coexistence of Humanity, in relation to nature.

G A I A T O K E N P R O J E C T T E A M

Nilton Konnupp
CEO and Software Programmer.

Social profile - LinkedIn:
https://www.linkedin.com/in/nilton-carlos-konnupp-4b0a1627/

Davi Konnupp
Administer Director and Promoter



Social profile - LinkedIn:
https://www.linkedin.com/search/results/all/?heroEntityKey=urn%3Ali%3Afsd_profile%3AACoAAD3ro9gBOHMqPvABaKEm7RcZxFB6HxKYZX4&keywords=davi%20d.&origin=RICH_QUERY_SUGGESTION&position=7&searchId=d3baf215-5c96-4303-9d48-b7ed10751a4e&sid=wEd

Diego Guerra
Director of project analysis and programming and finance analyst.

Perfil social LinkedIn:
https://www.linkedin.com/in/diego-guerra-maimone-3a4625239

GAIA
The mascot and inspirational project.