Platform Architecture

By AdaLink

Table Of Content

Introduction

This chapter serves as an overarching introduction to the protocols and architecture discussed in subsequent chapters. It outlines the foundational concepts and objectives that underpin the AdaLink platform's design and operation.

Platform Overview

AdaLink is a groundbreaking initiative designed to forge connections and foster collaborations within the vibrant Cardano ecosystem. Serving as a nexus for Cardano businesses, influencers, NFT founders, marketers, and large capital delegators “i.e. whales”, AdaLink aims to revolutionize how partnerships are formed and sustained within the blockchain space.

Motivation

AdaLink aims to create an ecosystem where businesses can advertise their own affiliate programs. The primary motivation is to foster growth and participation within Cardano by creating a marketplace for advertising affiliate and cooperative programs.

Key Objectives

1. **Automation**: Streamline processes through automation to reduce manual intervention and increase efficiency.
2. **Security**: Implement robust security measures to protect the integrity of the platform and user data.

With these objectives in mind, AdaLink has developed a series of protocols and architectural components to achieve its goals. The following chapters will delve into the specifics of these protocols, including logic flow, and how they integrate to form the platform's foundation.

Program Listing Protocol

Introduction

This Chapter goes over the protocol responsible to list affiliate programs onto the platform.

Protocol Overview

This protocol is the engine responsible for listing affiliate programs submitted by registered businesses. It ensures that all essential details of the program are collected from the businesses and stored in the database. The data is then made available to the frontend via APIs, allowing it to be displayed in the platform's designated section.

Protocol Components

Program ID

Each affiliate program gets assigned a unique ID code by AdaLink when they are listed on the platform. This ID will be used throughout the backend and frontend algorithms to reference each program uniquely.

Program Details

These are the related information that describes an affiliate program, e.g. commission rate, description field, project name, affiliate link, etc. The program's details are encoded inside the first output's datum within the "Listing Transaction" explained below.

Listing Transaction

Registering a new affiliate program is done by submitting a transaction on the blockchain by a registered business. This transaction includes two main parameters. First is the cost for listing, this consists of an amount of ADA. Second parameter is the affiliate program details "e.g. name, description, commission rate, affiliate link, etc.". These details are encoded as a datum that is attached to the transaction output.

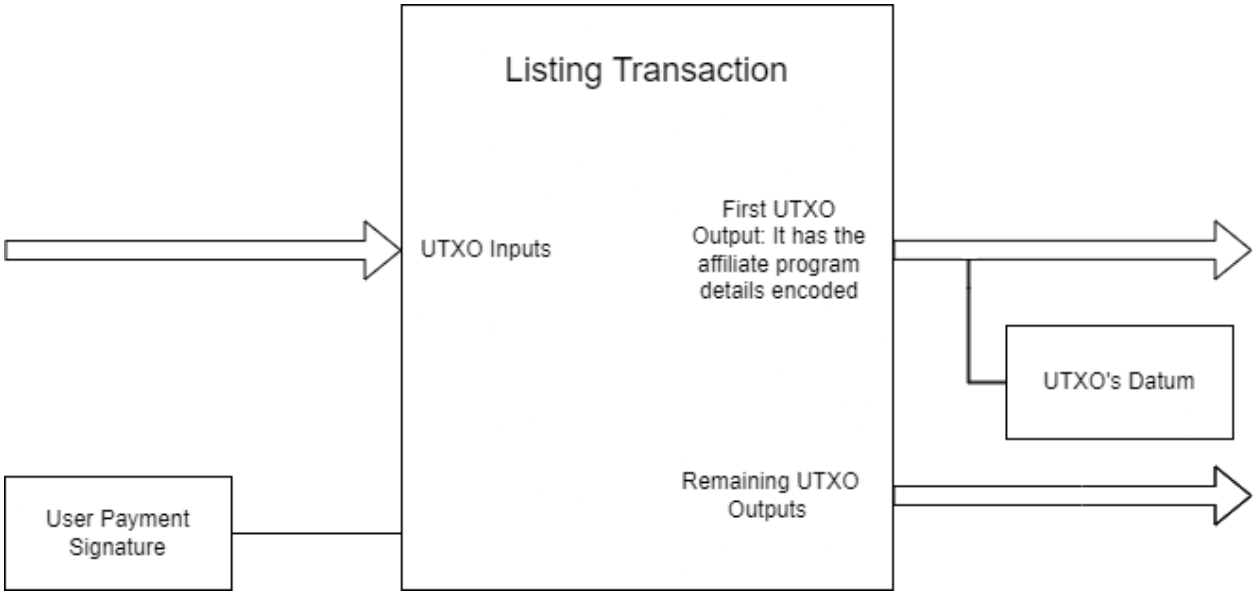

Figure 2-1: Listing Transaction Schematic.

Program Registrar

This is a running automated script in the backend of AdaLink. It constantly query the blockchain for new incoming “Listing Transactions”. Each time a new transaction is received, it decodes its datum and retrieves the affiliate program details from it. It then uploads these details into the server database.

Program Lister

AdaLink offers API for the frontend of the platform that is responsible to retrieve current active submitted affiliate programs and their details[1] from the database. The frontend then can translate these data and display each affiliate program on to the platform accordingly.

[1] These details are defined in chapter 3.

Protocol Logic Flow

1. Listing Request
    - A registered business creates a listing request for their affiliate program by submitting a “Listing Transaction”.
2. Affiliate Program Registration
    - “Program Registrar” script receives the listing request.
    - It decodes the affiliate program details. Then, it uploads them into the server database accordingly.
3. Affiliate Programs Retrieval
    - The platform’s frontend can, at any time, by the use of an API query all currently listed affiliate programs from the database.
    - It then displays them onto the platform.

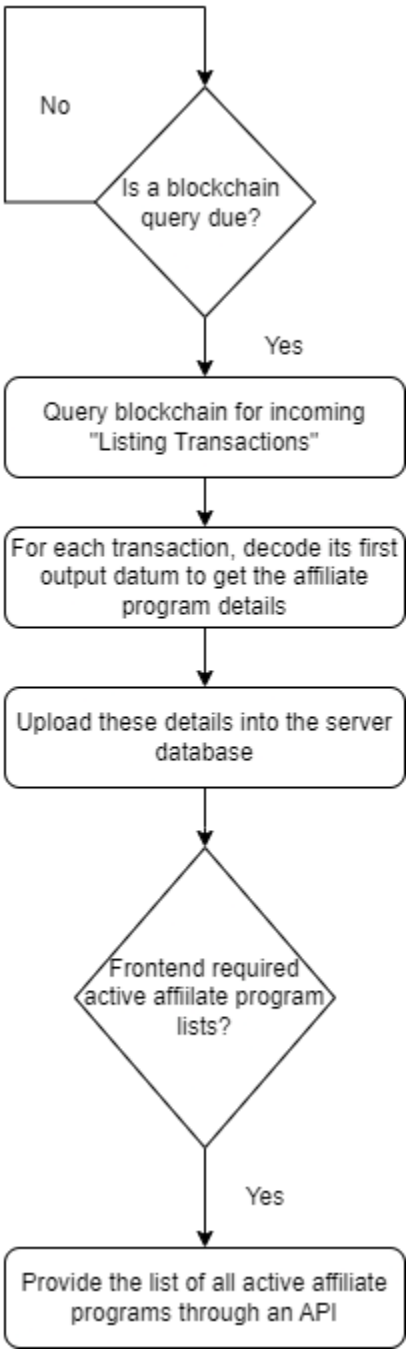

Figure 2-2: Flowchart of the program listing protocol.

Conclusion

The Program Listing Protocol lies at the heart of the AdaLink platform, designed to facilitate the submission and listing of affiliate programs by registered businesses. It operates by collecting essential program details through a blockchain-based transaction, which are then stored in a database. These details are accessible to the platform's frontend via APIs, ensuring that affiliate programs are properly displayed to users. The protocol's automated backend scripts handle the continuous monitoring of incoming transactions, decoding the data and updating the system. This comprehensive mechanism ensures seamless listing, registration, and retrieval of affiliate programs on the platform.

Database Architecture and Data Handling

Introduction

This chapter discusses how AdaLink deals with all types of data collected, stored and produced by the platform. After understanding the main goal of the project and how the major protocol operates, one can now lay down the architecture of the database used to store and retrieve previously acquired data. Additionally, the different sources for querying data from the blockchain is discussed as well.

Database Architecture

The data used by the platform is divided into 3 categories. Namely, businesses' data, affiliates' data and affiliate programs' data.

Businesses' Data

This database consists of a table that includes all related information about registered businesses. This information is adequate to describe the business or project to potential affiliates and users. The parameters and their data types are the following:

- Display Name (Static, but editable by user): STRING date type.
- ID (Static): STRING data type.
- Profile Picture (Static, but editable by user): URL data type.
- Type: Possible options are, DEFI, marketplace, NFT project (Static): STRING data type.
- Description (Static, but editable by user): STRING data type.
- Website (Static, but editable by user): URL data type.
- Social Link (Static, but editable by user): URL data type.
- Registered Wallet Address (Static): STRING data type.
- Registered Wallet Stake Address (Static): STRING data type.

ID	Display Name	Type	Description	Wallet Address	Stake Address	Profile Picture	Website	Social Link

Figure 3-1: Illustration of the businesses' table format within the platform database.

Affiliates' Data

This database consists of a table that includes all related information about registered affiliates. The parameters and their data types are the following:

- Unique ID (Static): STRING data type.
- Display Name (Static): STRING data type.
- Type: Possible options are, influencer, web3 project, marketing agency (Static): STRING data type.
- Registered Wallet Address (Static): STRING data type.
- Registered Wallet Stake Address (Static): STRING data type.
- Profile Picture (Static, but editable by user): URL data type.
- Website (Optional): URL data type.
- Social Link (Optional): URL data type.

ID	Display Name	Type	Wallet Address	Stake Address	Profile Picture	Website	Social Link

Figure 3-2: Illustration of the Affiliates' table format within the platform database.

Affiliate Programs' Data

This database consists of a table that includes all affiliate programs' information, both active and ended ones. This information is adequate to describe an affiliate program for interested affiliates and users. The parameters and their data types are the following:

- Affiliate Program ID (Static): STRING data type.
- Affiliate Program Name (Static): STRING data type.
- Affiliate Link (Static): URL data type.
- Project Name (Static): STRING data type.
- Project Website (Static): URL data type.
- Start Date (Static): POSIX data type.
- End Date (Static): POSIX data type.
- Program's Commision Rate (Static): FLOAT data type.

With these parameters some useful metrics can be deducted and shown/rendered on the platform as well. Some of these metrics are:

- Program's Status: values are active or ended.
- Program's Remaining Time: Can be shown for active programs.

ID	Program Name	Affiliate Link	Project Name	Project Website	Start Date	End Date	Rate

Figure 3-3: Illustration of the affiliate programs' table format within the platform database.

Data Collection

All data stored in the database are acquired from two sources. First one is user input, mostly during registration. Second source is the blockchain ledger itself.

Data Acquired from Users' Input

Typically these data are for users to introduce themselves to other users on the platform. No data from this type can result in any unwanted changes in the core outcomes from any of the protocols. These parameters are mostly collected during the registration stage. The following data parameters are examples of this type of data:

- Project's Display Name.
- Project's Profile Picture.
- Project's Type.
- Project's Description.
- Project's Website.
- Project's Social Link.
- Affiliate's Display Name.
- Affiliate's Type.
- Affiliate's Profile Picture.
- Affiliate's Website.
- Affiliate's Social Link.

Data Handling Scripts Overview

The approach of collecting these types of data is done using PHP scripts in the backend server. These scripts are called by the frontend, making them local APIs, where they fill in or update the database with the data fed to them by the user. The logic flow for collecting this type is as follows:

- The user fills in some fields in the frontend, i.e. the platform webpage.
- The user submits their input values, mostly through a confirmation button.
- This frontend packs the inserted data and query the specified API(s) with this data.
- The script(s) in the backend depackage the incoming data and store them in their respective location within the platform's database.
- A feedback sent back to the frontend either confirming the process or reporting an error and requests a re-do.

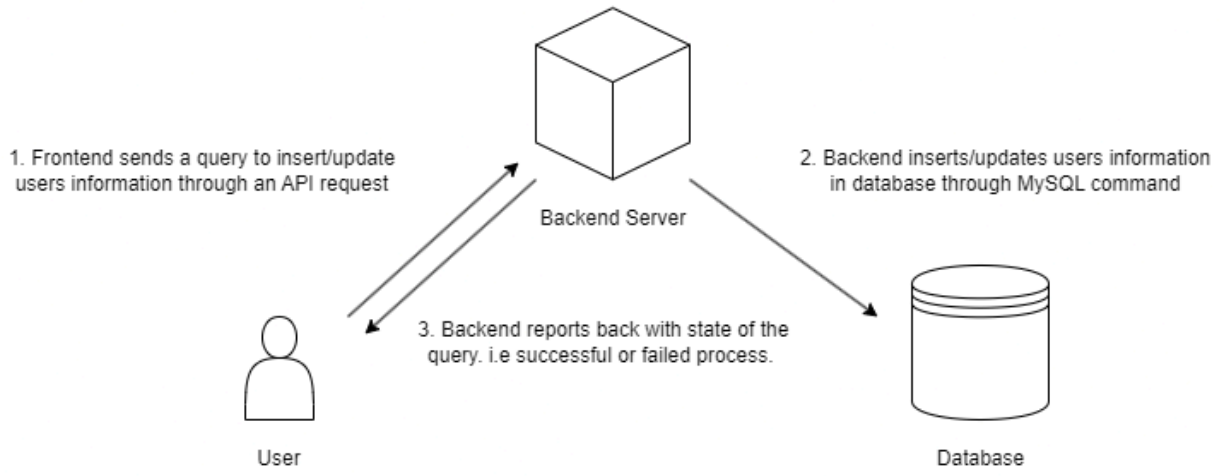

Figure 3-4: Illustration of handling the data acquired by users' inputs.

Data Acquired from the Blockchain

This data contains the majority of the information used on the platform. Adalink has two main gateways to query data from the blockchain. First one is the direct query from a full node installed in the backend server of the platform. Second source is Blockfrost API. All remaining data discussed in the previous section is retrieved by an API call in the backend or by directly querying the data from the installed node.

Data Handling Scripts Overview

This data type is retrieved from the blockchain via shell scripts within the backend and stored in the database. Later on, the stored/collected data can be acquired by both the backend protocols (i.e. via shell scripts) and the frontend web page (i.e. via PHP scripts).

Data Collecting Scripts

This is the stage of retrieving data from the blockchain. As stated earlier, this is done via shell scripts run within the backend. The logic flow can be summarized as follows:

- The script is triggered, i.e. by a timer event or external trigger.
- The script query either the installed node by using CLI commands or via API calls sent to Blockfrost database.
- Upon successful retrieval, the data is stored in its respective location within the platform's database.

Data Query Scripts - for Backend

These are shell scripts and are parts of the protocols that run in the backend. Their main purpose is to retrieve needed data from the database and use them in the protocol.

Data Query Scripts - for Frontend

These are PHP scripts that are triggered by the frontend through API calls. The scripts query the desired data from the database and send it back to the user. The logic flow of this process is below:

- The frontend requires a set of data via an API call.
- The API call triggers the corresponding script in the backend.
- The script pulls out the desired data values from the database.
- The script sends back the values of the desired data to the frontend.

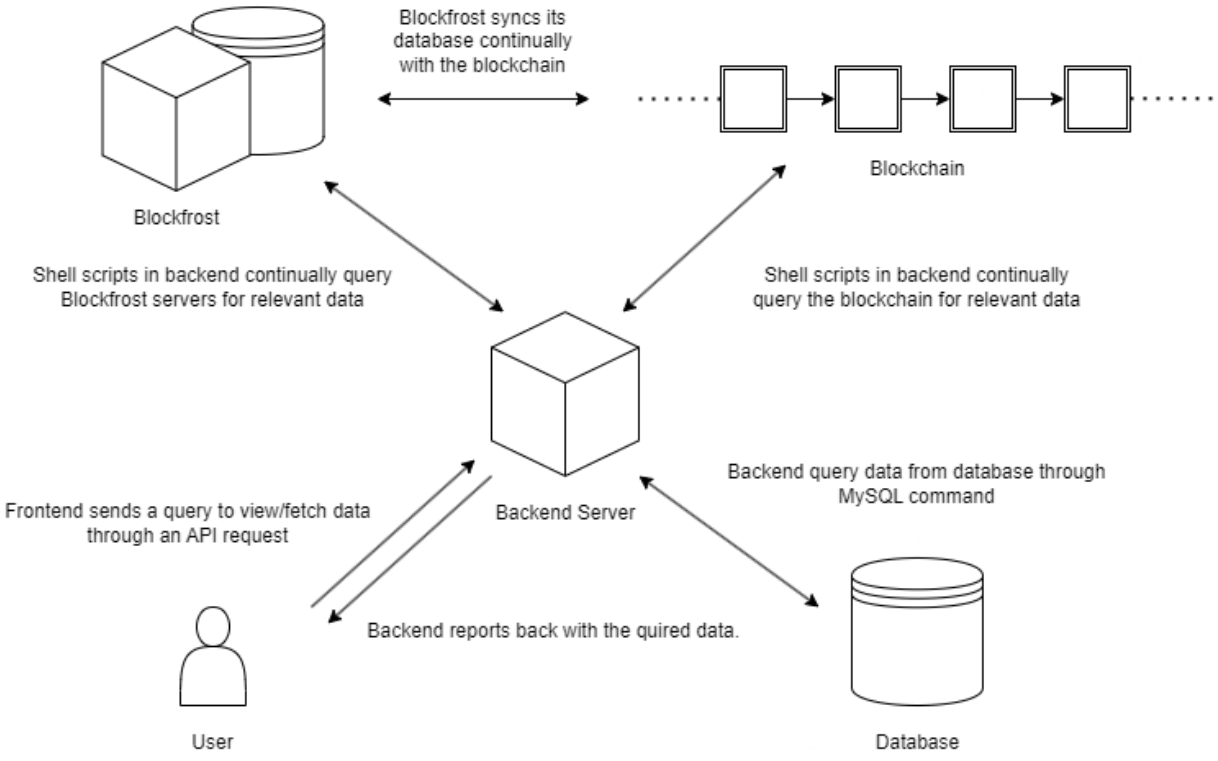

Figure 3-5: Illustration of handling the data acquired from the blockchain.

Conclusion

AdaLink's database architecture and data handling protocols provide a structured framework for managing projects, businesses, affiliates, and affiliate programs data efficiently. Through user input during registration and blockchain queries, the platform collects essential information. Backend shell scripts and PHP scripts facilitate data handling and retrieval, ensuring seamless communication between frontend and backend components. This robust system establishes AdaLink as a reliable and scalable platform.