PoE Direct

Introduction

PoE Direct is a website that is meant to service the crafting community of Path of Exile. It exists to help buy sell and trade services for Path of Exile such as boss kills, harvest crafts and mirroring items. Currently Path of Exile doesn't have a publically facing trade website avaliable for these types of services and many users are required to use Discord or another platform to conduct their trades leading to negative interactions, fractured communities and major portions of the player base not being interested in interacting with these systems. PoE direct is a website that should simplify this process and make it more accessible to the PoE community at large.

Requirements

- Authenticates users with their Path of Exile.com account
- Lists trades publically for users
- Supports different leagues
- Supports multiple categories of services/trades
- Supports Vouching for Crediability
- Supports Reporting on Crediability

Design

Introduction

The overall design of PoE Direct is similar to most modern web applications users will be served up a javascript bundle from a CDN which connects to backend API servers that populate and read data out of a backend datastore. The overall flow is made simple throught he use of GraphQL for the API access layer and managed services from AWS to host the infrastructure. Cost is an important factor with this design as this web service will be operated on donations and advertisements.

Front End Design

< INSERT CONTENT HERE>

API Design

Types

These are the type definitions for GraphQL and represent the concrete types of data that will be returned from our GraphQL API.

Category

- **ID**
  - **Type** - int64
- **Name**
  - **Type** - string

Service

- **ID**
  - **Type** - int64
- **Name**
  - **Type** - string
- **CategoryID**
  - **Type** - int64

Trade

- **ID**
  - **Type** - int64
- **LeagueID**
  - **Type** - int64
- **ServiceID**
  - **Type** - int64
- **UserID**
  - **Type** - int64
- **Price**
  - **Type** - int32
- **Currency**
  - **Type** - Enum String

Vouch

- **ID**
  - **Type** - int64
- **TradeID**
  - **Type** - int64

- **BuyerID**
  - **Type** - int64
- **SellerID**
  - **Type** - int64
- **Message**
  - **Type** - string

Report

- **ID**
  - **Type** - int64
- **BuyerID**
  - **Type** - int64
- **SellerID**
  - **Type** - int64
- **Message**
  - **Type** - string

Queries

This is a list of queries for the GraphQL API, queries are GET type operations that let us retrieve data structured as types from our DB based on a set of parameters.

- **GetCategory**
  - **Input**
    - **CategoryID**
      - **Type** - int64
  - **Return**
- **GetCategories**
  - **Input**
    - nil
  - **Return**
    - **Type** - Array Category
- **GetService**
  - **Input**
    - **ServiceID**
      - **Type** - int64
  - **Return**
    - **Type** - Service
- **GetServices**
  - **Input**
    - **CategoryID**
      - **Type** - int64
  - **Return**

        - **Type** - Array Service
- **GetTrade**
    - **Input**
        - **ID**
            - **Type** - int64
    - **Return**
        - **Type** - Trade
- **GetTrades**
    - **Input**
        - nil
    - **Return**
        - **Type** - Array Trade
- **GetVouch**
    - **Input**
        - **ID**
            - **Type** - int64
    - **Return**
        - **Type** - Vouch
- **GetVouches**
    - **Input**
        - nil
    - **Return**
        - **Type** - Array Vouch
- **GetReport**
    - **Input**
        - **ID**
            - **Type** - int64
    - **Return**
        - **Type** - Report
- **GetReports**
    - **Input**
        - nil
    - **Return**
        - **Type** - Array Report

Mutations

Mutations are PUT operations that let us mutate or create data in our database through our GraphQL API.

- **PutCategory**
    - **Input**

        - **Name**
            - **Type** - String
    - **Return**
        - **Type** - Category
- **PutService**
    - **Input**
        - **Name**
            - **Type** - String
        - **CategoryID**
            - **Type** - int64
    - **Return**
        - **Type** - Service
- **PutTrade**
    - **Input**
        - **LeagueID**
            - **Type** - int64
        - **ServiceID**
            - **Type** - int64
        - **Price**
            - **Type** - int32
        - **Currency**
            - **Type** - Enum String
    - **Return**
        - **Type** - Trade
- **PutVouch**
    - **Input**
        - **UserID**
            - **Type** - int64
        - **TradeID**
            - **Type** - int64
        - **Message**
            - **Type** - String
    - **Return**
        - **Type** - Vouch
- **PutReport**
    - **Input**
        - **UserID**
            - Type - int64
        - **TradeID**
            - Type - int64
        - **Message**
            - Type - String
    - **Return**

    - **Type** - Report

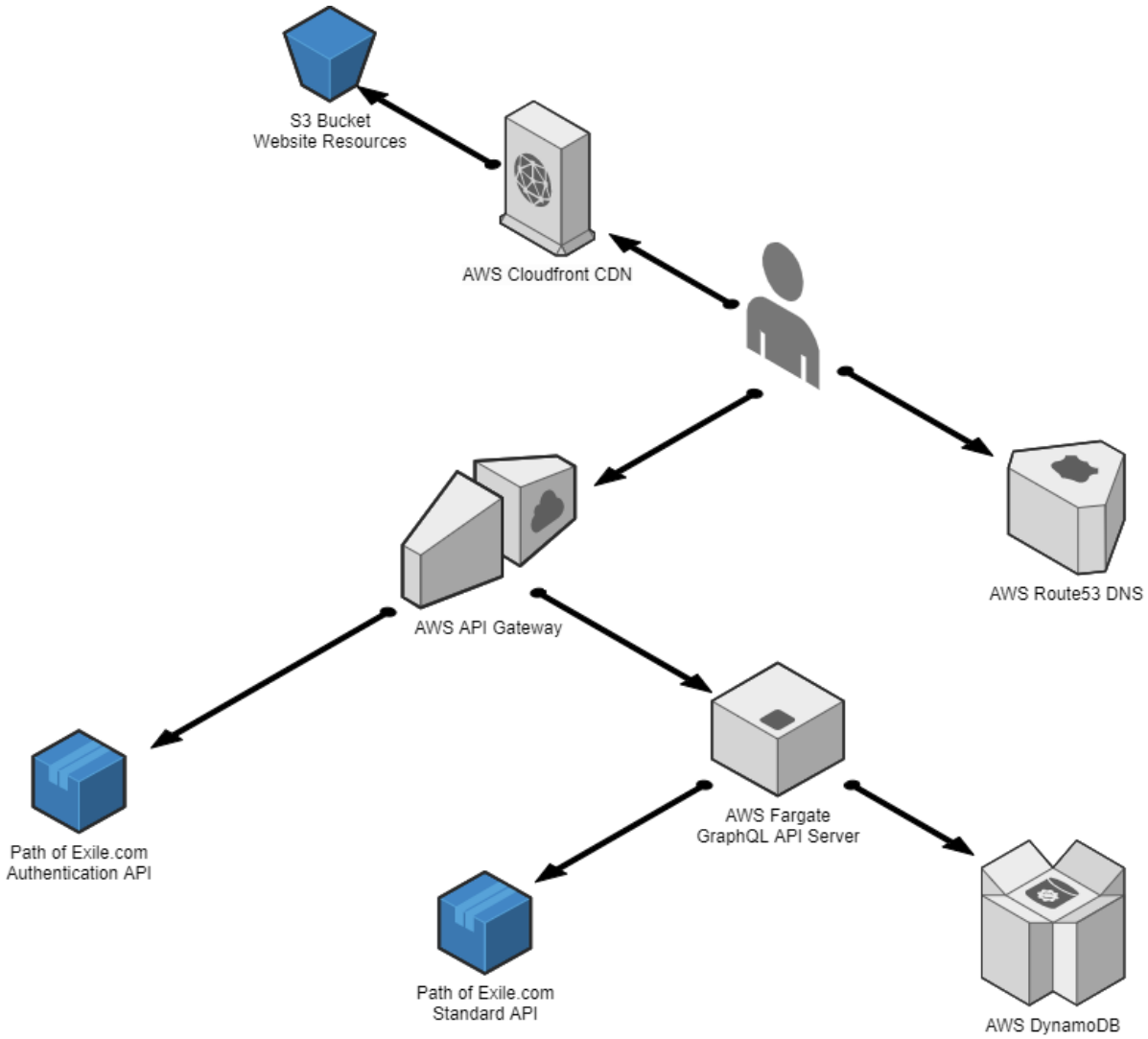
Architecture Design

Introduction

Terms

- **AWS Route53** - Used for DNS registration
- **AWS API Gateway** - Used for routing and authentication for API calls
- **AWS Cloudfront** - Used to distribute website resources JS, HTML, CSS etc...
- **AWS S3** - Used to store website resources JS, HTML, CSS, etc...
- **AWS Fargate** - Used to host API server
- **AWS DynamoDB** - Used to store user/trade information
- **AWS Cloudwatch** - Used for monitoring/logging/alarming/etc...

Diagram



Flow

1. User attempts to browse to <insert_url_here_for_site> Route53 responds with API Gateway Endpoint
2. API Gateway Determines if Users is Authenticated if the user is authenticated skip to 4 if not continue.
3. API Gateway authenticates user using OAuth2 whith PathOfExile.com
4. API Gateway forwards request to GraphQL API server running on AWS Fargate Instance
5. Website Queries/Submissions are actioned by AWS Fargate GraphQL server
6. GraphQL server interactes with PathOfExile.com's API as well as DDB database tables to action queries and then responds to the users browser.

Backend Schema Design

Users Table

The users DB will hold a list of PoE ID's, SessionIDs, Status and Vouch/Reports for a particular user.

- **UserID**
  - **Type** - int64
  - **Value** - "The users POE userID from pathofexile.com"
  - **Unique** - Yes
- **UserName**
  - **Type** - str
  - **Value** - "The users name from pathofexile.com"
  - **Unique** - Yes
- **UserStatus**
  - **Type** - enum string
  - **Value** - "The users current status from a string enum of different valid status's"
  - **Unique** - False
- **UserVouches**
  - **Type** - Array int64
  - **Value** - "A list of vouch IDs from the vouches table for this particular user"
  - **Unique** - No
- **UserReports**
  - **Type** - Array int64
  - **Value** - "A list of user report IDs from the reports table for this particular user"
  - **Unique** - No
- **CharacterName**
  - **Type** - str
  - **Value** - "The users currently logged in character name from pathofexile.com"
  - **Unique** - No

Vouches Table

A table that holds all vouches

- **VouchID**
  - **Type** - int64
  - **Value** - "A uniquely generated ID for each vouch"
  - **Unique** - Yes
- **BuyerUserID**
  - **Type** - int64
  - **Value** - "The value of the buyer that this vouch was made by"
  - **Unique** - Yes
- **SellerUserID**
  - **Type** - int64
  - **Value** - "The value of the seller that this vouch was made for"
  - **Unique** - Yes
- **TradeID**
  - **Type** - int64
  - **Value** - "The ID that this trade was made for"

    - **Unique** - Yes
- **VouchMessage**
    - **Type** - string
    - **Value** - "The message that was added to this vouch by the vouching user"
    - **Unique** - No

Reports Table

A table that holds all reports

- **ReportID**
    - **Type** - int64
    - **Value** - "A uniquely generated ID for each report"
    - **Unique** - Yes
- **BuyerUserID**
    - **Type** - int64
    - **Value** - "The value of the buyer that this report was made by"
    - **Unique** - Yes
- **SellerUserID**
    - **Type** - int64
    - **Value** - "The value of the seller that this report was made for"
    - **Unique** - Yes
- **ReportMessage**
    - **Type** - string
    - **Value** - "The message that was added to this report by the reporting user"
    - **Unique** - No

Categories Table

A table that holds all categories

- **CategoryID**
    - Type - int64
    - Value - "Uniquely generated key that represents a particular category"
    - Unique - True
- **CategoryName**
    - **Type** - str
    - **Value** - "Name of a particular category I.E. - Harvest, Boss Kill, Mirror Service, ETC..."
    - **Unique** - False

Services Table

The services table will hold relevant information regarding a service's category, name and ID.

- **CategoryID**
    - **Type** - int64

    - **Value** - "Uniquely generated key that represents a particular category from the category table I.E. Harvest Craft/Boss Kill/Mirror Service/ETC"
    - **Unique** - True
- **ServiceID**
    - **Type** - int64
    - **Value** - "Uniquely generated key for a particular type of service I.E. Harvest craft add fire remove fire"
    - **Unique** - True
- **ServiceName**
    - **Type** - str
    - **Value** - "A name of a particular service I.E. Add/Remove Fire does not need to indicate category"
    - **Unique** - False

Trades Table

Schema for the trade table looks like the following,

- **TradeID**
    - **Type** - int64
    - **Value** - "Uniquely generated trade key for every trade posting"
    - **Unique** - True
- **LeagueID**
    - **Type** - int64
    - **Value** - "Uniquely generated key for each league"
    - **Unique** - True
- **UserID**
    - **Type** - int64
    - **Value** - "Uniquely generated UserID that comes from pathofexile.com"
    - **Unique** - True
- **ServiceID**
    - **Type** - int64
    - **Value** - "Uniquely generated ServiceID that can be used to lookup service information in Services table"
    - **Unique** - False
- **Price**
    - **Type** - int32
    - **Value** - "A value set by a user to determine what the price of a trade will be"
    - **Unique** - False
- **Currency**
    - **Type** - STR/ENUM
    - **Value** - "One of the various allowed string identifiers for what type of currency the price is for I.E. exalt/chaos/chrome/alt/etc..."
    - **Unique** - False
- **Status**
    - **Type** - Bool
    - **Value** - "Active or Inactive to determine if a trade has been completed or not or is still active"

- **Unique** - False