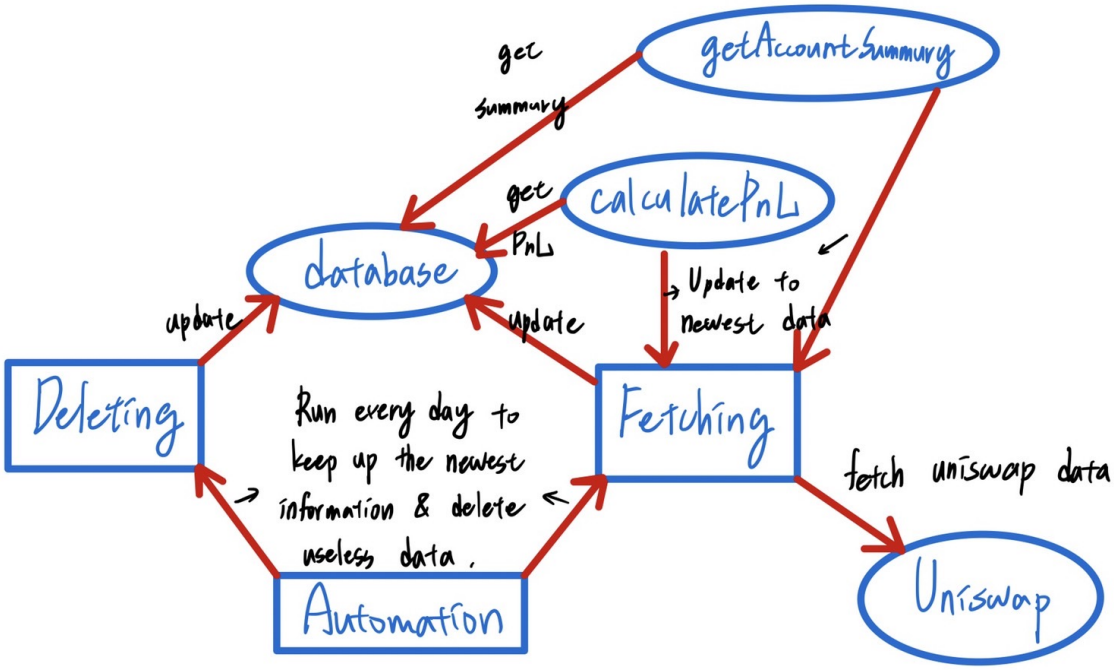
# System Design

## Diagram of the design




- **Fetching Function(private)**: It would take the argument $address$, and find the last updated time, and query Uniswap API to fetch the trade information after the time then update the information stored in database.
- **Deleting Function(private)**: It would check all the Trade older than 2 years, clean them up and caculate the new value then update the information stored in database.
- **Automation element(private)**: It would find all addresses from database then call **Fetching Function** of these addresses every day to update the data of the address. It would also call **Deleting Function** to clean up data older than 2 years.
- **getAccountSummary API(public)**: It would call **Fetching Function** first, and then query database to return the data needed by user.

- **calculatePnL API(public)**: It would call **Fetching Function** first, then return the PnL from database.

## API Document

- getAccountSummary:
  - Endpoint: /summary/\<address>
  - Method: GET
  - Parameters:
    - address: Address of the user to retrieve summary data for
  - Response:

```
{
    "address": "0x416db23fa4eac3264f431666b701a85d08973e51",
    "trades": 100,
    "tokens": ["ETH", "USDC"],
    "most_traded_token": "USDC"
}
```

- calculatePnL:
  - Endpoint: /pnl/\<address>
  - Method: GET
  - Parameters:
    - address: Address of the user to calculate PnL for
  - Response:

```
{
    "address": "0x416db23fa4eac3264f431666b701a85d08973e51",
    "pnl": 1234.56
}
```

## Database design

- Account:

```
type Account {
    id: int
    address: string
    trades: int # number of trades
    tokens: TokenRecord[] # tokens traded
    mostTradedToken: TokenRecord # most traded token
    totalPnL: float
    updated: Date # last updated time
}
```

  When ever fetching function fetch the data, it would calculate new values of trades, tokens, mostTradedToken, and totalPnL in **Account** and **TokenRecord** for the minimal latency requirement.

- Trade: All information of trades.

```
type Trade {
    id: int
    account: Account
    tokenIn: Token
    tokenOut: Token
    tradeDate: Date
}
```

- Token: This is for token traded in one transaction.

```
type Token {
    id: int
    account: Account
    name: string
    symbol: string
    amount: float
    amountUSD: float
}
```

- TokenRecord: This for total token traded by an address.

```
 type TokenRecord {
    id: int
    account: Account
    name: string
    symbol: string
    trades: int # total number of trades of the token
    amount: float # total amount traded
}
```