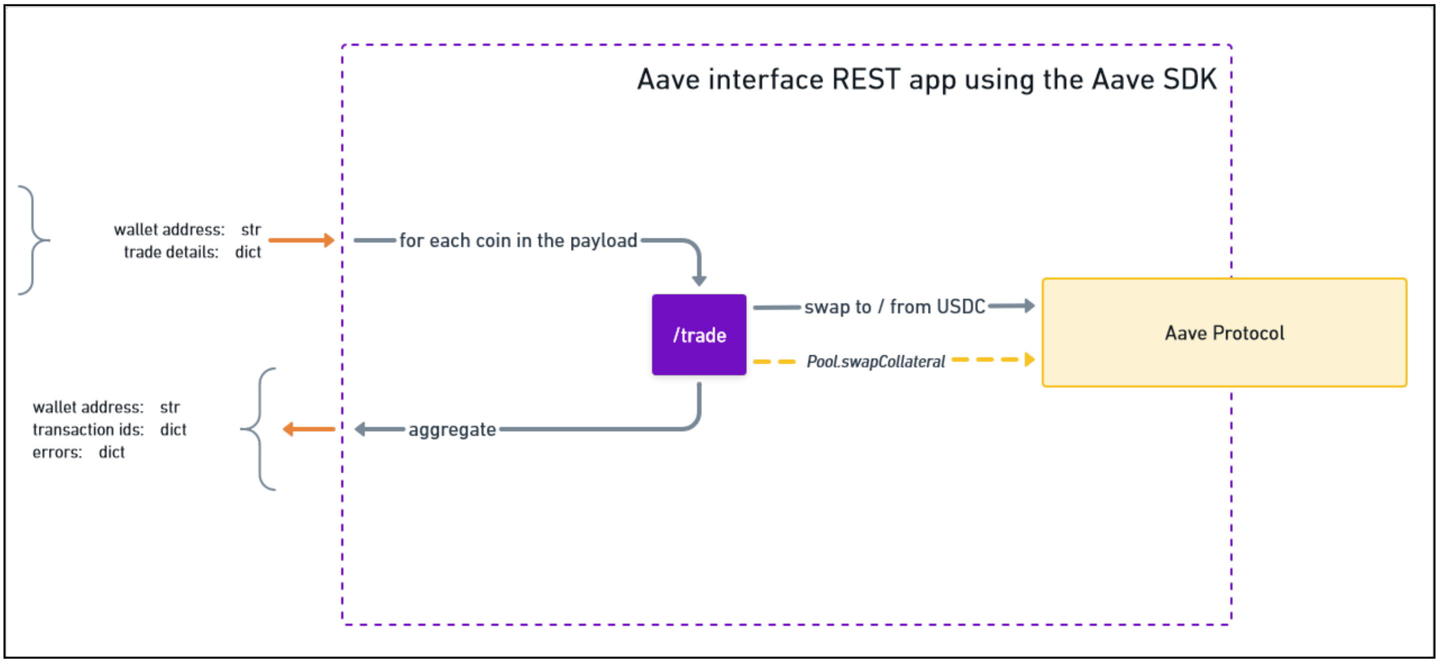

Market Interface (REST API) for Aave

- Written in TypeScript.
- Use Node.js 16.x LTS.
- Use pnpm instead of npm/yarn.
- Use Prettier for automatic code formatting, with a line length of 88.

- A freestanding REST app, running a HTTP/2 capable server that handles concurrency / asynchronous requests well.
- Will use the Aave SDK (https://github.com/aave/aave-utilities#pool-v3).
  - It should use the Polygon Pool.
- Accepts a POST request at route /trade, with a msgpack-ed (https://msgpack.org) payload containing the amounts of each coin to swap:

<table>
<tr><td>wallet address</td><td>string</td></tr>
<tr><td>trade details</td><td>dictionary<table><tr><td>coin ticker (string)</td><td>amount to trade (float)</td></tr></table></td></tr>
</table>

for ex.

```
{
  "wallet_address": "0x0000000000000000000000000000000000000000",
  "trade_details": {
    "ETH": -5.233,
    "BTC": 0,
    "MATIC": 5.899
  }
}
```

  - If the delta for a given coin is +, collateral should be swapped for the coin.
  - If the delta for a given coin is -, the coin should be swapped for collateral.
  - If the delta for a given coin is 0, no trade should happen with that coin.

- After all trades are submitted and transactions are returned (or errored), the route should return a msgpack-ed payload containing the wallet address, transaction ids for each trade, or any errors that prevented a transaction from being created.
  - *Is it possible that the Aave SDK would error before submitting a transaction to the blockchain? If yes, that is what `errors` would be used for.*

<table>
<tr><td>wallet address</td><td>string</td></tr>
<tr><td>transaction ids</td><td>dictionary<table><tr><td>coin ticker (string)</td><td>transaction id (string)</td></tr></table></td></tr>
<tr><td>errors</td><td>dictionary<table><tr><td>coin ticker (string)</td><td>transaction error</td></tr></table></td></tr>
</table>

for ex.

```
{
  "wallet_address": "0x0000000000000000000000000000000000000000",
  "transaction_ids": {
    "ETH": "0x0000000000000000000000000000000000000000",
    "MATIC": "0x0000000000000000000000000000000000000000"
  },
  "errors": {}
}
```

- Write unit and integration tests.
  - The integration tests should mock an incoming HTTP POST request with some arbitrary wallet and trade details, and should mock calls to the actual Aave protocol (or use the testnet).