Secret Gift Exchange

Write a microservice to manage members of the gift exchange and assign recipients.

There are many many different ways to solve this problem. What we are looking for is simply good code and good architecture. It's up to you to decide what that is.

We want your solution to be simple enough to complete in a few hours (it doesn't necessarily have to be "perfect" if there is such a thing). Find a good medium, write the best code you can. If you don't have time to finish send what you have, just explain what you planned on doing.

If you get done and decide you could have done something better, feel free to add commentary on what you would have done differently if you were to do it again.

Problem Description

Imagine that an extended family (parents, children, grandparents, aunts/uncles, cousins, etc) does a yearly gift exchange, in which each family member draws another person's name at random, and then buys a gift for that person. Write a program that will assign each family member another family member to give a gift to. Each family member should only receive one gift.

Constraints

A person cannot be assigned to give a gift to themselves.
A family member cannot be given a gift from the same person more than once every 3 years.

Project Setup

- You'll need git installed to complete this project. Please install it if you haven't already.
- Create a new local git repository for this project (git init)
- When you're done, create a git bundle to send to us (e.g. *git bundle create giftexchange.bundle master*)

Tips

- Your code should be easily readable and maintainable
- Include unit tests to validate your application's functionality (edge cases, normal operation, etc)
- If you need to store data, using an in-memory "database" is OK (simple Collections are fine).
- You do not need a datastore to persist data across application restarts, but think about how it

would be modeled in a datastore if you were to transition from your application's in-memory datastore to a persistent datastore.
● Your application may be viewed by many family members at the same time. Even though your application is single threaded, you may want to consider what would happen with concurrent access.
● You do not need a UI, just unit tests.

Minimal API

Data

Family Member

```
Unset
{
      "id": string, "name": string

}
```

Gift Exchange

```
Unset
{
      "member_id": string, "recipient_member_id": string
}
```

REST

- GET /members -- list the family members
- GET /members/{id} -- get a single family member
- POST /members -- add a family member
- PUT /members/{id} -- updates a family member
- DELETE /members/{id} -- delete a family member
- GET /gift_exchange -- lists members along with the member id they will be gifting to

An API will be needed to shuffle the gift exchange and has been left as an exercise.