Plano de Estudos

Passo 1: Testes Unitários

- https://www.devmedia.com.br/teste-unitario-com-junit/41235
- https://www.devmedia.com.br/junit-tutorial/1432
- https://medium.com/@gcbrandao/testando-uma-api-rest-spring-boot-2-com-junit5-e-mockmvc-db603c65a306
- https://www.udemy.com/course/testes-com-junit-5-mockito-e-spring-boot-rest-apis/

Passo 2: Conhecimentos de Spring/Hibernate

Ler:

- http://luizricardo.org/2014/03/strings-em-java-ha-mais-detalhes-do-que-voce-imagina/
- https://www.devmedia.com.br/autoboxing-e-unboxing-em-java/28620
- https://www.devmedia.com.br/conhecendo-as-classes-wrappers-autoboxing-e-auto-unboxing/7384
- https://blog.algaworks.com/entendendo-o-equals-e-hashcode/
- https://blog.cod3r.com.br/desmistificando-hashcode-e-equals-em-java/
- https://angeliski.com.br/equals-e-hashcode?x-host=angeliski.com.br
- https://www.devmedia.com.br/sobrescrevendo-o-metodo-hashcode-em-java/26488
- https://www.devmedia.com.br/padrao-de-projeto-factory-method-em-java/26348
- https://www.thiengo.com.br/padrao-de-projeto-factory-method
- https://www.youtube.com/watch?v=-PT-pXe-7UM
- https://www.devmedia.com.br/definindo-beans-de-sessao-singleton-em-java/28358
- https://www.devmedia.com.br/lazy-e-eager-loading-com-hibernate/29554

Responder as perguntas abaixo:

========== Java Core ==========

- O resultado de `"batata" == "batata"` é `true` ou `false`? Porque?
- O resultado de `"batata".equals("batata")` é `true` ou `false`? Porque?
- O resultado de `new String("batata") == new String("batata")` é `true` ou `false`? Porque?
- O resultado de `new String("batata").equals(new String("batata"))` é `true` ou `false`? Porque?

- O que é boxing?
- O que é autoboxing
- O que é unboxing?
- O que é o método `equals`? qual a função dele?
- O que é o método `hashcode`? qual a função dele?
- Qual o resultado da comparação abaixo?

```
public class Batata {
        private int id;
}

private batata1 = new Batata(1);
private batata2 = new Batata(1);

batata1 == batata2 -> true ou false?
```

- Qual o resultado da comparação abaixo?

```
public class Batata {
        private int id;
}

private batata1 = new Batata(1);
private batata2 = new Batata(1);

batata1.equals(batata2) -> true ou false?
```

- Qual o resultado da comparação abaixo?

```
public class Batata {
        private int id;

        public int getId() {
                return this.id;
        }

        @Override
        public boolean equals(Object other) {
            if (this == other) {
                return true;
                }

            if (other == null || getClass() != other.getClass()) {
```

```
                return false;
                }

            Batata batata = (Batata) other;

            return this.getId() == batata.getId();
        }
}

private batata1 = new Batata(1);
private batata2 = new Batata(1);

batata1.equals(batata2) -> true ou false?
```

- Qual o resultado da comparação abaixo?

```
public class Batata {
        private int id;

        public int getId() {
                return this.id;
        }

        @Override
        public boolean equals(Object other) {
            if (this == other) {
                return true;
                }

            if (other == null || getClass() != other.getClass()) {
                return false;
                }

            Batata batata = (Batata) other;

            return this.getId() == batata.getId();
        }
}

private batata1 = new Batata(1);
private batata2 = new Batata(2);

batata1.equals(batata2) -> true ou false?
```

- Qual o resultado da comparação abaixo?

```
public class Batata {
        private int id;

        public int getId() {
                return this.id;
        }

        @Override
        public boolean equals(Object other) {
            if (this == other) {
                return true;
                }

            if (other == null || getClass() != other.getClass()) {
                return false;
                }

            Batata batata = (Batata) other;

            return this.getId().equals(batata.getId());
        }
}

Set<Batata> batatas = new HashSet<>();
batatas.add(new Batata(1));
batatas.add(new Batata(1));

batatas.size() -> 1 ou 2?
```

- Qual o resultado da comparação abaixo?

```
public class Batata {
        private int id;

        public int getId() {
                return this.id;
        }

        @Override
        public boolean equals(Object other) {
            if (this == other) {
                return true;
                }

            if (other == null || getClass() != other.getClass()) {
```

```
                return false;
                }

            Batata batata = (Batata) other;

            return this.getId().equals(batata.getId());
        }

        @Override
        public int hashCode() {
            return Objects.hash(id);
        }
}

Set<Batata> batatas = new HashSet<>();
batatas.add(new Batata(1));
batatas.add(new Batata(1));

batatas.size() -> 1 ou 2?
```

========== Spring ==========

- Qual o escopo padrão de um bean do spring?
- O que é um bean `singleton`?
- O que é uma variavel de estado em um bean?
- Qual o perigo de modificar o estado de um bean `singleton`?
- Quando é seguro modificar o estado de um bean?
- O que acontece na classe abaixo se ela for chamada várias vezes de forma concorrente?

```
@Service
@Singleton
public class Batata {
        private int counter = 0;

        public int incrementAndGet() {
                this.counter = this.counter + 1;
                return counter;
        }
}
```

========== Hibernate ==========

- O que é Hibernate `Eager`?
- O que é Hibernate `Lazy`?
- Qual é o padrão de uma Entidade do hibernate, `Eager` ou `Lazy`?
- Como criar uma query JPQL? Dê um exemplo.

========== Factory Pattern ==========

- Crie uma classe para converter arquivos de vários tipos (xml, html, txt, csv, json) para um modelo interno usando o padrão factory. A `factory` deverá receber como input o tipo de arquivo e devolver qual a classe `converter` adequada para o mesmo.

========== HackerHank ==========

Resolver os problemas abaixo:

- https://www.hackerrank.com/challenges/solve-me-first/problem?isFullScreen=true
- https://www.hackerrank.com/challenges/simple-array-sum/problem?isFullScreen=true
- https://www.hackerrank.com/challenges/compare-the-triplets/problem?isFullScreen=true
- https://www.hackerrank.com/challenges/a-very-big-sum/problem?isFullScreen=true
- https://www.hackerrank.com/challenges/diagonal-difference/problem?isFullScreen=true
- https://www.hackerrank.com/challenges/plus-minus/problem?isFullScreen=true
- https://www.hackerrank.com/challenges/staircase/problem?isFullScreen=true
- https://www.hackerrank.com/challenges/mini-max-sum/problem?isFullScreen=true
- https://www.hackerrank.com/challenges/birthday-cake-candles/problem?isFullScreen=true
- https://www.hackerrank.com/challenges/time-conversion/problem?isFullScreen=true
- https://www.hackerrank.com/challenges/grading/problem?isFullScreen=true
- https://www.hackerrank.com/challenges/apple-and-orange/problem?isFullScreen=true
- https://www.hackerrank.com/challenges/kangaroo/problem?isFullScreen=true
- https://www.hackerrank.com/challenges/between-two-sets/problem?isFullScreen=true
- https://www.hackerrank.com/challenges/breaking-best-and-worst-records/problem?isFullScreen=true
- https://www.hackerrank.com/challenges/the-birthday-bar/problem?isFullScreen=true
- https://www.hackerrank.com/challenges/divisible-sum-pairs/problem?isFullScreen=true
- https://www.hackerrank.com/challenges/migratory-birds/problem?isFullScreen=true
- https://www.hackerrank.com/challenges/day-of-the-programmer/problem?isFullScreen=true
- https://www.hackerrank.com/challenges/bon-appetit/problem?isFullScreen=true
- https://www.hackerrank.com/challenges/sock-merchant/problem?isFullScreen=true
- https://www.hackerrank.com/challenges/drawing-book/problem?isFullScreen=true

- https://www.hackerrank.com/challenges/counting-valleys/problem?isFullScreen=true
- https://www.hackerrank.com/challenges/cats-and-a-mouse/problem?isFullScreen=true

Passo 3: Projeto Apam Api

Api Apam

Crie uma api para gerenciar a APAM. A Api deverá conter os endpoints abaixo, com os métodos de adicionar/atualizar/buscar/deletar aplicáveis a cada endpoint, realizando as mudanças necessárias no banco.

A arquitetura deverá seguir a seguinte regra:

- Controller: Onde ficará a configuração do endpoint
- Service: Onde ficará a logica de negocio
- Model: Modelos utilizados pelo Service
- Repository: Acesso ao banco de dados
- Entity: Entidades do Banco de dados

O caminho deverá ser sempre `Controller -> Service (Model) -> Repository (Entity)`, sendoq ue o controler só vai chamar o service para realizar as operações e retornar o resultado e o service irá utilizar o repositorio para realizar as operações.

As classes de entity e repository não podem ser utilizadas no Controller, somente classes do Model ou service.

Todo o código deverá ser feito em inglês e ter testes unitários e de integração, como no exemplo de endpoint `/health` contido no repositório.

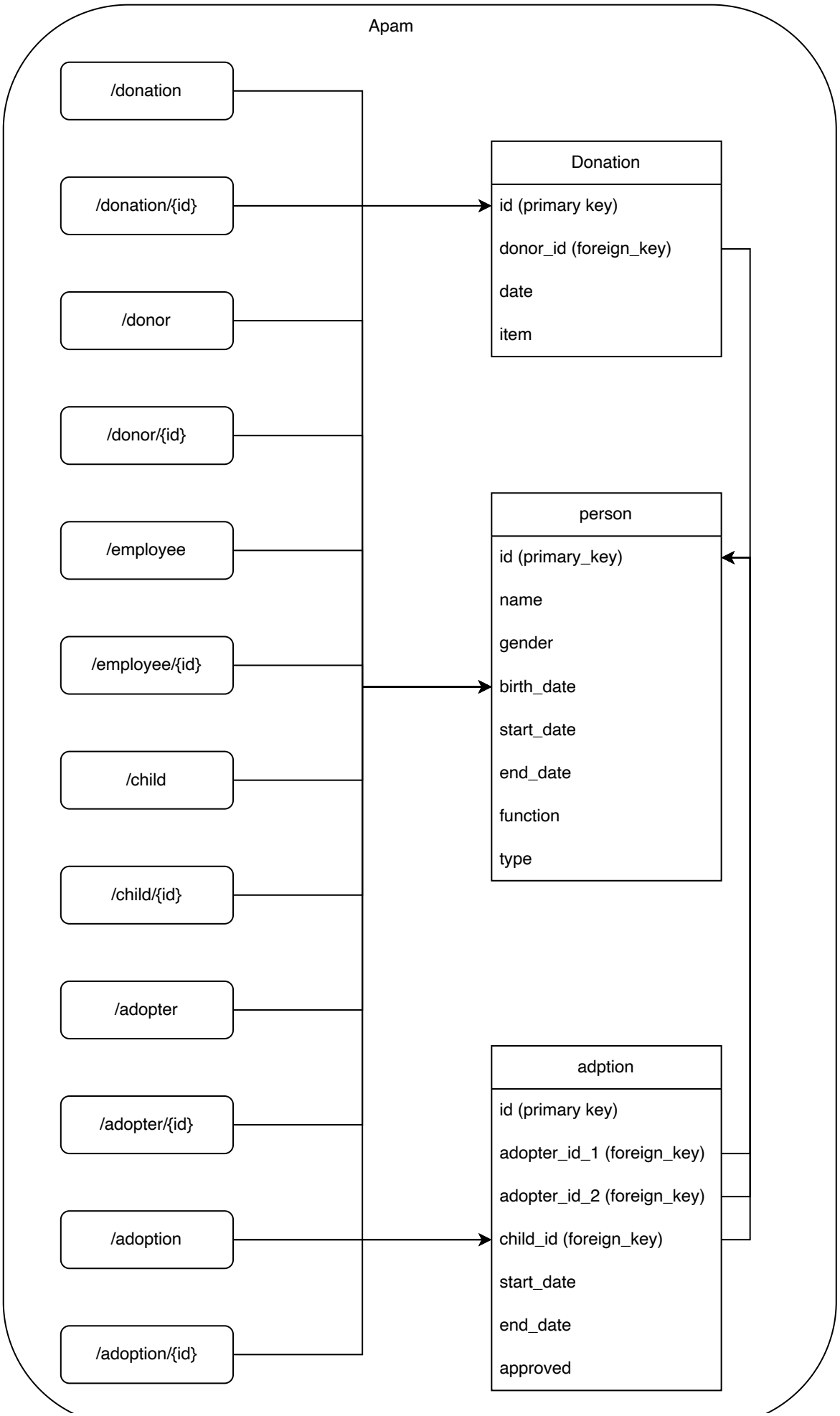
As tabelas deverão ser criadas usando migrações do flyway.

Repository: https://github.com/dawsonfi/apam-api
Convite: https://github.com/dawsonfi/apam-api/invitations

- Endpoints
  - /child
    - Add/Update/Find/Delete
  - /child/{id}
    - Update/Find/Delete
  - /employee
    - Add/Update/Find/Delete
  - /employee/{id}

        - Update/Find/Delete
    - /donor
        - Add/Update/Find/Delete
    - /donor/{id}
        - Update/Find/Delete
    - /donation
        - Add/Update/Find/Delete
    - /donation/{id}
        - Update/Find/Delete
    - /adopter
        - Add/Update/Find/Delete
    - /adopter/{id}
        - Update/Find/Delete
    - /adoption
        - Add/Update/Find/Delete
    - /adoption/{id}
        - Update/Find/Delete
- Database Tables
    - person
        - id (primary key)
        - name
        - gender (can be null)
        - birth_date
        - start_date
        - end_date (can be null)
        - function: (can be null)
        - type
            - child
            - employee
            - donor
            - adopter
    - donation
        - id (primary key)
        - donor_id: (foreign_key) -> person
        - date: date
        - item: string
    - adoption

- id (primary key)
- adopter_id_1: (foreign_key) -> person
- adopter_id_2: (foreign_key) -> person (can be null)
- child_id: (foreign_key) -> person
- start_date: date
- end_date: date (can be null)
- approved: boolean


Apam
/donation
/donation/{id}
/donor
/donor/{id}
/employee
/employee/{id}
/child
/child/{id}
/adopter
/adopter/{id}
/adoption
/adoption/{id}
Donation
id (primary key)
donor_id (foreign_key)
date
item
person
id (primary_key)
name
gender
birth_date
start_date
end_date
function
type
adption
id (primary key)
adopter_id_1 (foreign_key)
adopter_id_2 (foreign_key)
child_id (foreign_key)
start_date
end_date
approved