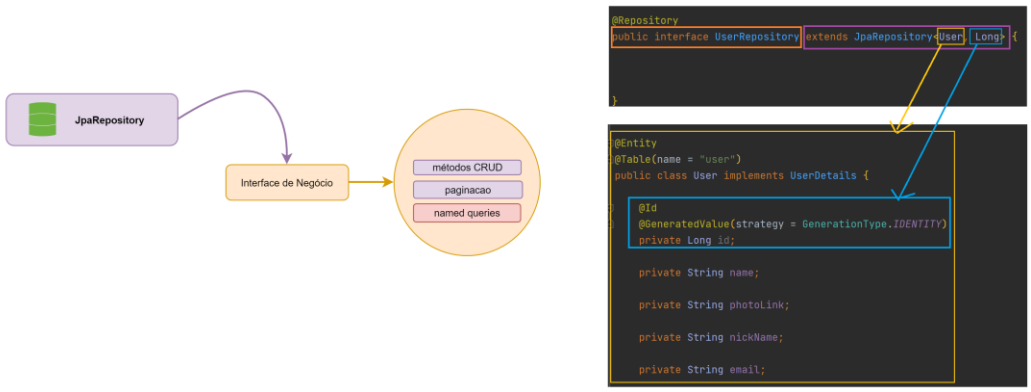
Named Queries - Spring Data JPA

Um dos recursos mais interessantes que o **SDJ** fornece são as named queries, basicamente falando são queries que podem ser escritas através apenas da ***assinatura de um método***.

Tudo que precisamos fazer é **criar uma interface** que estenda de **alguma interface do JPA** *(nesse caso foi a própria **JpaRepository<T,K>** que herda de todas as interfaces do sdj)*, informando o **modelo mapeado** e o **tipo da chave primária**, a partir daí temos uma interface de repositório capaz de usufruir de tudo que o SDJ oferece:



A **minha interface** que herdou da JpaRepository já pode fazer muita coisa, como por exemplo **métodos de CRUD**. Mas existem situações onde eu quero uma **query específica**, por exemplo, achar um usuário pelo e-mail dele, e é aí que as **named queries** brilham.

O **SDJ** fornece uma série de **prefixos (que representam uma palavra chave em queries de banco de dados)** que podem ser combinados ao nome dos **atributos das nossas entidades** para que **queries sejam criadas com jpql**:

Keyword	Sample	JPQL snippet
And	findByLastnameAndFirstname	… where x.lastname = ?1 and x.firstname = ?2
Or	findByLastnameOrFirstname	… where x.lastname = ?1 or x.firstname = ?2
Is,Equals	findByFirstname,findByFirstnameIs,findByFirstnameEquals	… where x.firstname = 1?
Between	findByStartDateBetween	… where x.startDate between 1? and ?2
LessThan	findByAgeLessThan	… where x.age < ?1
LessThanEqual	findByAgeLessThanEqual	… where x.age <= ?1
GreaterThan	findByAgeGreaterThan	… where x.age > ?1
GreaterThanEqual	findByAgeGreaterThanEqual	… where x.age >= ?1
After	findByStartDateAfter	… where x.startDate > ?1

Por exemplo, se eu quisesse encontrar um usuário pelo email dele eu criaria uma **assinatura de método** que segue o conceito da named querie:

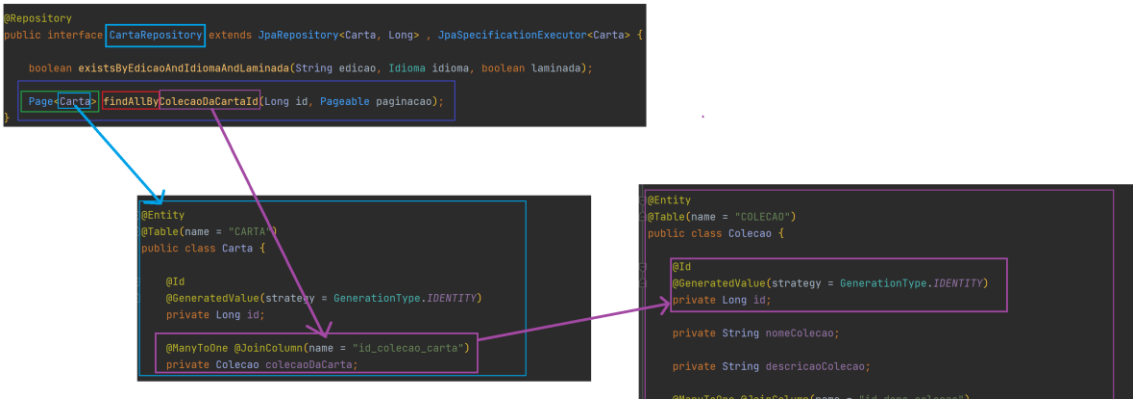
```
@Repository
public interface UserRepository extends JpaRepository<User, Long> {

    Optional<User> findByEmail(String email);

}
```

Mas não pense que **named queries** são somente pra coisas tão bobinhas assim *(embora se encaixem muito melhor nelas do que em queries complexas).*

Em um cenário que eu tenho **um projeto onde usuários podem mostrar suas cartas de magic através de coleções** eu poderia por exemplo, **encontrar todas as cartas de uma determinada coleção**, retornando o **resultado paginado**. A **named querie** foi poderosa o suficiente para entender uma **querie com entidades relacionadas:**



A documentação completa das **named queries** podem ser acessada aqui: https://docs.spring.io/spring-data/jpa/docs/1.5.0.RELEASE/reference/html/jpa.repositories.html