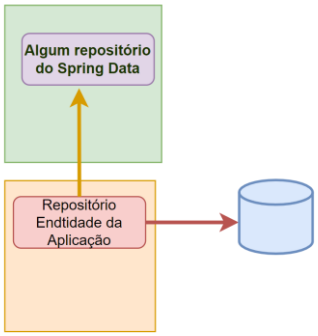
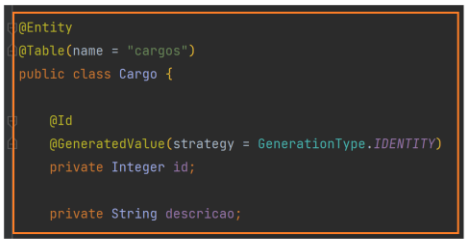
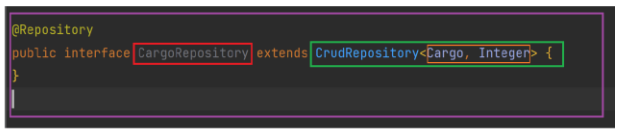
Uma vez **que o projeto já tenha a dependência** do **Spring Data JPA**, basta usar suas **interfaces**, o jeito mais simples de se trabalhar com **interfaces** de **repositório do SDJ** é fazer a **interface** de **repositório da aplicação** estender de alguma **Interface do SDJ:**



Por exemplo, eu tenho uma **Entidade Cargo** na minha **aplicação**, ela **vai se comunicar com o banco de dados usando a Interface CargoRepository**, minha **CargoRepository vai herdar da interface CrudRepository** do **Spring Data** que já tem os métodos de **CRUD** implementados:

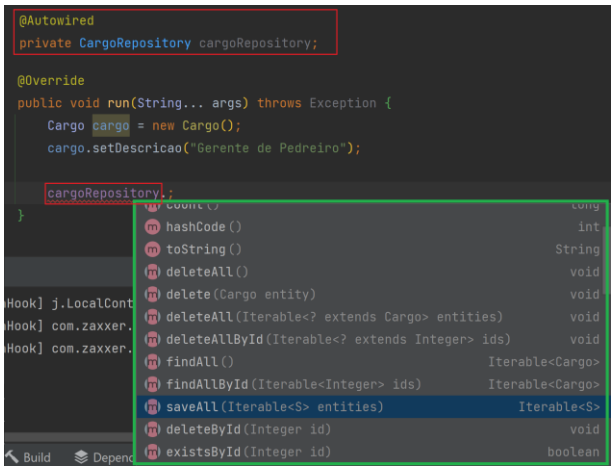


```java
@Entity
@Table(name = "cargos")
public class Cargo {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    private String descricao;
```



```java
@Repository
public interface CargoRepository extends CrudRepository<Cargo, Integer> {
}
```

Agora a partir do meu **CargoRepository** eu posso usar qualquer método do **CrudRepository do SDJ** que serve para a **Entidade informada na herança** (Cargo):



```java
@Autowired
private CargoRepository cargoRepository;

@Override
public void run(String... args) throws Exception {
    Cargo cargo = new Cargo();
    cargo.setDescricao("Gerente de Pedreiro");

    cargoRepository.;
}
```

```
hashCode()                                              int
toString()                                              String
deleteAll()                                             void
delete(Cargo entity)                                    void
deleteAll(Iterable<? extends Cargo> entities)           void
deleteAllById(Iterable<? extends Integer> ids)          void
findAll()                                               Iterable<Cargo>
findAllById(Iterable<Integer> ids)                      Iterable<Cargo>
saveAll(Iterable<S> entities)                           Iterable<S>
deleteById(Integer id)                                  void
existsById(Integer id)                                  boolean
```