Bootcamp Crisalis

Orange Team

Faccipieri, Federico

Fleitas, Gonzalo

Flores, Elio

Haoys, Sebastián

Pepa Degani, Julián

CRUD de Clientes (backend)

Responsable: Faccipieri, Federico
JIRA : SCRUM-21 BackEnd

Descripción

Se creó el CRUD en el backend siguiendo las normativas del lenguaje JAVA. A través del mismo se podrá crear, editar, listar y eliminar clientes (sean empresas o personas físicas).

Para diferenciar respectivamente entre una persona física como cliente o bien, una empresa, se utilizaron 2 enrutados, o *endpoints*, diferentes: "../person" y "../enterprise" respectivamente.

Para realizar esta lógica se utilizaron modelos (o entidades), así como clases controladoras, DTOs, servicios y repositorios.

Requerimientos

- "../enterprise/create" es el endpoint para crear un cliente de tipo empresa.
- "../enterprise/edit" es el endpoint para editar un cliente de tipo empresa.
- "../enterprise/disable" es el endpoint para eliminar un cliente de tipo empresa.
- "../enterprise/getAll" es el endpoint para listar todos los clientes de tipo empresa.

- "../person/create" es el endpoint para crear un cliente de tipo persona.
- "../person/edit" es el endpoint para editar un cliente de tipo persona.
- "../person/disable" es el endpoint para eliminar un cliente de tipo persona.
- "../person/getAll" es el endpoint para crear un cliente de tipo persona.

Entidades

En las siguientes imágenes pueden observarse las declaraciones de las tres entidades antes referidas, así como sus respectivos parámetros.

```java
@Entity
public class ClientEntity {

    2 usages
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO, generator = "name")
    @GenericGenerator(name="name", strategy="native")
    private int id;

    3 usages
    @NotNull
    private boolean beneficiary;
```

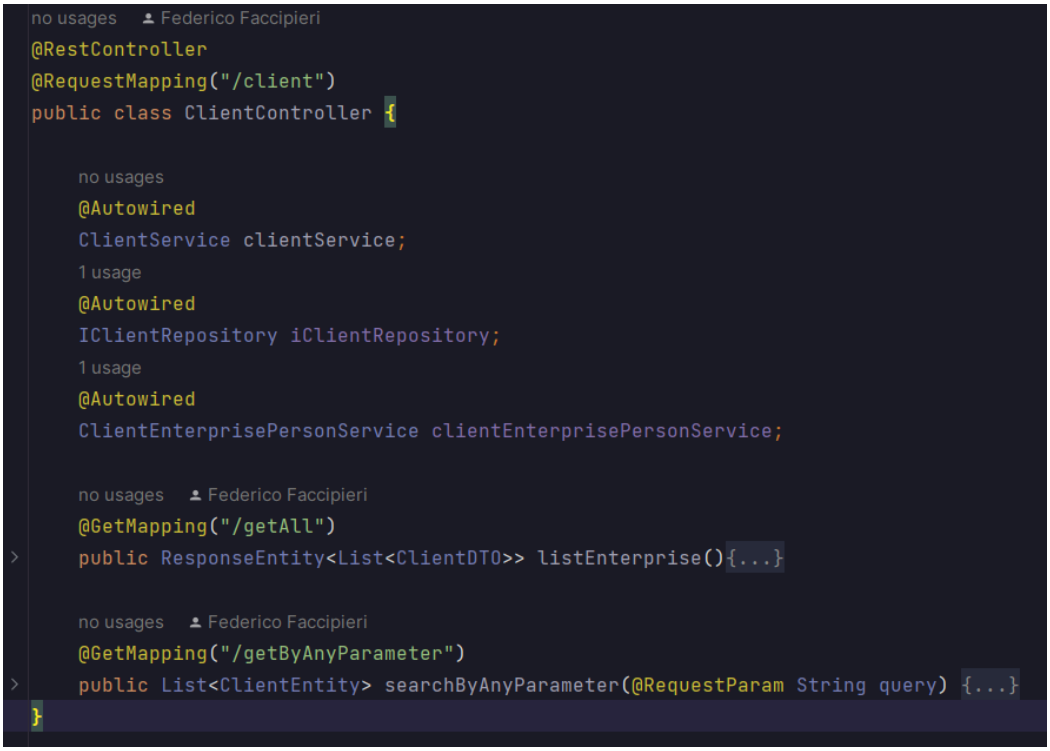
Entidad cliente (ClientEntity)

Entidad Persona (PersonEntity)

Entidad empresa (EnterpriseEntity)

Controladores

En la siguiente imagen se observa el controlador de la Entidad Cliente donde al final se observa el método searchByAnyParameter que refiere a la query realizada para poder implementar búsquedas dinámicas.



A continuación se observa una captura del controlador de “Enterprise” o empresa, en donde pueden observarse las declaraciones de los diferentes métodos y sus respectivos *endpoints* ya mencionados.

En esta otra imagen se observa algo muy similar a la anterior, con la salvedad de que lo que se observa es todo lo referido a la persona cliente.

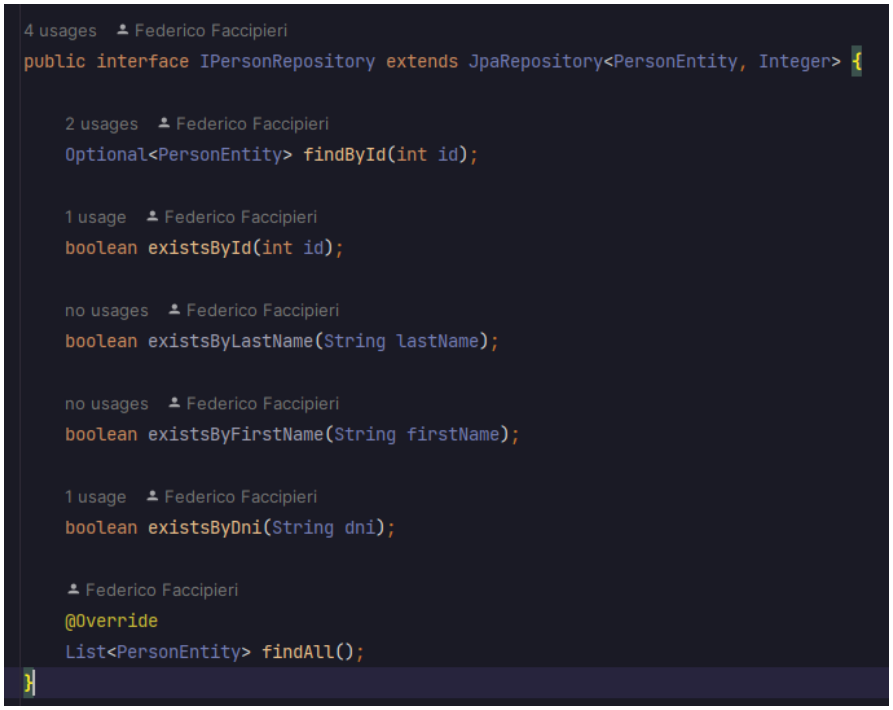
```java
@RestController
@RequestMapping("/person")
public class PersonController {

    2 usages
    @Autowired
    PersonService personService;

    5 usages
    @Autowired
    IPersonRepository iPersonRepository;

    no usages  Federico Faccipieri *
    @PostMapping("/create")
    public ResponseEntity<?> create(@Valid @RequestBody NewPersonDTO newPersonDTO, BindingResult bindingResult){...}

    no usages  Federico Faccipieri
    @GetMapping("/getAll")
    public ResponseEntity<List<PersonDTO>> listPerson(){...}

    no usages  Federico Faccipieri
    @PostMapping("/edit")
    public ResponseEntity<?> editPersonHandler(@Valid @RequestBody PersonDTO personDTO, BindingResult bindingResult){...}

    no usages  Federico Faccipieri
    @PostMapping("/disable")
    public ResponseEntity<?> disable(@Valid @RequestBody PersonDTO personDTO, BindingResult bindingResult){...}
}
```

Repositorios

En las siguientes imágenes pueden observarse los códigos Java referidos a los repositorios de las entidades.

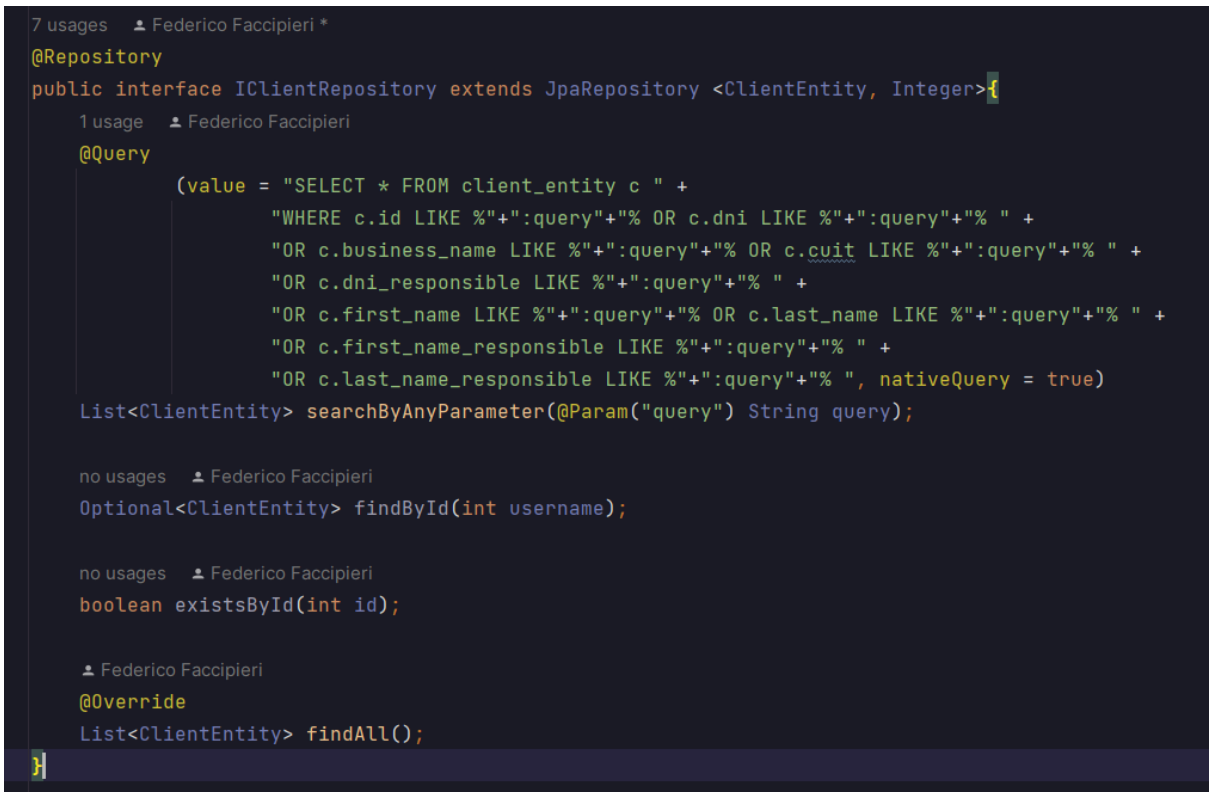
```java
4 usages  Federico Faccipieri
public interface IEnterpriseRepository extends JpaRepository<EnterpriseEntity, Integer> {

    2 usages  Federico Faccipieri
    Optional<EnterpriseEntity> findById(int id);

    1 usage  Federico Faccipieri
    boolean existsById(int id);

    1 usage  Federico Faccipieri
    boolean existsByCuit(String cuit);

    1 usage  Federico Faccipieri
    boolean existsByBusinessName(String businessName);

     Federico Faccipieri
    @Override
    List<EnterpriseEntity> findAll();
}
```

Repositorio de Empresa



```
4 usages  Federico Faccipieri
public interface IPersonRepository extends JpaRepository<PersonEntity, Integer> {

    2 usages  Federico Faccipieri
    Optional<PersonEntity> findById(int id);

    1 usage  Federico Faccipieri
    boolean existsById(int id);

    no usages  Federico Faccipieri
    boolean existsByLastName(String lastName);

    no usages  Federico Faccipieri
    boolean existsByFirstName(String firstName);

    1 usage  Federico Faccipieri
    boolean existsByDni(String dni);

    Federico Faccipieri
    @Override
    List<PersonEntity> findAll();
}
```

Repositorio de Persona



```
7 usages  Federico Faccipieri *
@Repository
public interface IClientRepository extends JpaRepository <ClientEntity, Integer>{
    1 usage  Federico Faccipieri
    @Query
            (value = "SELECT * FROM client_entity c " +
                    "WHERE c.id LIKE %"+":query"+"% OR c.dni LIKE %"+":query"+"% " +
                    "OR c.business_name LIKE %"+":query"+"% OR c.cuit LIKE %"+":query"+"% " +
                    "OR c.dni_responsible LIKE %"+":query"+"% " +
                    "OR c.first_name LIKE %"+":query"+"% OR c.last_name LIKE %"+":query"+"% " +
                    "OR c.first_name_responsible LIKE %"+":query"+"% " +
                    "OR c.last_name_responsible LIKE %"+":query"+"% ", nativeQuery = true)
    List<ClientEntity> searchByAnyParameter(@Param("query") String query);

    no usages  Federico Faccipieri
    Optional<ClientEntity> findById(int username);

    no usages  Federico Faccipieri
    boolean existsById(int id);

    Federico Faccipieri
    @Override
    List<ClientEntity> findAll();
}
```

Repositorio de Cliente

Servicios

Finalmente, en estas últimas imágenes, podremos ver el código utilizado en los servicios.

```
3 usages  Federico Faccipieri
@Service
@Transactional
public class EnterpriseService {

    4 usages
    @Autowired
    IEnterpriseRepository iEnterpriseRepository;

    no usages  Federico Faccipieri
    public boolean existsById(int id) { return iEnterpriseRepository.existsById(id); }

    1 usage  Federico Faccipieri
    public boolean existsByCuit(String cuit) { return iEnterpriseRepository.existsByCuit(cuit); }

    2 usages  Federico Faccipieri
    public boolean existsByBusinessName(String businessName){
        return iEnterpriseRepository.existsByBusinessName(businessName);
    }

    1 usage  Federico Faccipieri
    public void save(EnterpriseEntity enterpriseEntity) { iEnterpriseRepository.save(enterpriseEntity); }
}
```

Servicios de Empresa

```
2 usages  Federico Faccipieri *
@Service
@Transactional
public class PersonService {

    3 usages
    @Autowired
    IPersonRepository iPersonRepository;

    no usages  Federico Faccipieri
    public boolean existsById(int id) { return iPersonRepository.existsById(id); }

    1 usage  Federico Faccipieri
    public boolean existsByDni(String dni) { return iPersonRepository.existsByDni(dni); }

    1 usage  Federico Faccipieri
    public void save(PersonEntity personEntity) { iPersonRepository.save(personEntity); }
}
```

Servicios de Persona

Query

Finalmente se realizó una query a modo de filtro para poder buscar a partir de cualquiera de los parámetros dentro de las tres entidades ya mencionadas. De este modo, se creó un modo de búsqueda dinámico. Para ello se utilizó el Repositorio ya creado de la Entidad Cliente y también se usó el Controlador de la Entidad Cliente. Por otro lado, se creó un nuevo Servicio llamado “ClientEnterprisePersonService” ya que es un servicio que involucra a las tres entidades.

A continuación puede observarse el código utilizado en dicho servicio.

Servicios de la Query