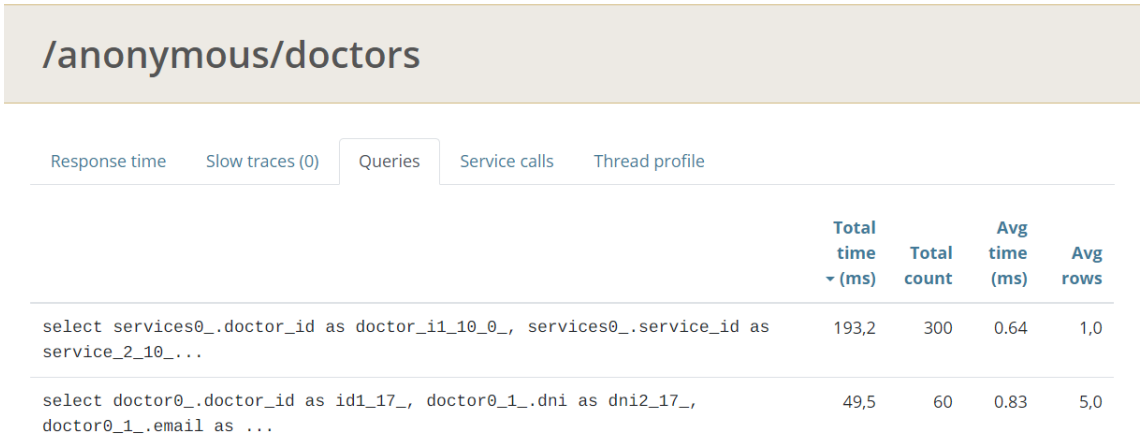
Profiling

- HU-001: Listado de los médicos de un centro

Hemos seleccionado esta historia ya que, el pico de usuarios más pequeño en el que se produce el cuello de botella en la aplicación entre nuestras historias, se corresponde con esta.

Una vez aplicada la carga de trabajo a la aplicación de unos 40 usuarios anónimos y unos 20 doctores, se obtienen los siguientes tiempos, para las queries utilizadas en el controlador DoctorController:



Como podemos observar, para cada uno de los doctores que se listan en la consulta, estamos consultando también los servicios que ofrecen cada uno de estos. Por lo que una de las soluciones para ahorrarnos estas consultas cada vez que consultamos los doctores de la base de datos, sería el traernos los servicios de estos cuando nos los traemos, haciendo así dos consultas en una. Para ello, tendremos que hacer una serie de cambios en diferentes partes de la aplicación:

1. Crear la nueva consulta que vamos a realizar, poniendo la nueva cabecera en la clase DoctorRepository y su implementación en SpringDataDoctorRepository:

ExaminationRepository.java | DoctorRepository.java | *SpringDataDoctorRepository.java

```java
package org.springframework.clinicaetsii.repository;

import java.util.Collection;

import org.springframework.clinicaetsii.model.Doctor;
import org.springframework.clinicaetsii.model.Service;
import org.springframework.dao.DataAccessException;
import org.springframework.dao.DataIntegrityViolationException;
import org.springframework.data.repository.query.Param;

public interface DoctorRepository {

    Collection<Doctor> findAll() throws DataAccessException;

    Collection<Doctor> findDoctorsSortedByNumOfServices() throws DataAccessException;

    Doctor findDoctorByUsername(String username) throws DataAccessException;

    Doctor findById(int id) throws DataAccessException;

    Doctor findDoctorById(int id) throws DataAccessException;

    Doctor findDoctorByPatientId(@Param("id") int id) throws DataAccessException;

    Doctor save(Doctor doctor) throws DataAccessException, DataIntegrityViolationException;

    Collection<Service> findAllServices() throws DataAccessException;

    void delete(Doctor d) throws DataAccessException;

    void deleteAll() throws DataAccessException;

    Collection<Doctor> findDoctorsWithServices() throws DataAccessException;
}
```

```java
@Transactional(readOnly = true)
public Collection<Doctor> findAllDoctorsWithServices() throws DataAccessException {
    return this.doctorRepository.findDoctorsWithServices();
}
```

2. Llamar en su controlador correspondiente al nuevo servicio creado.

```java
@GetMapping(value = "/anonymous/doctors")
public String processFind(final Doctor doctor, final BindingResult result, final Map<String, Object> model) {

    Collection<Doctor> doctors = this.doctorService.findAllDoctorsWithServices();
    if (doctors.isEmpty()) {
        model.put("emptylist", true);
    } else {
        model.put("doctors", doctors);

    }
    return "/anonymous/doctors/doctorsList";
}
```

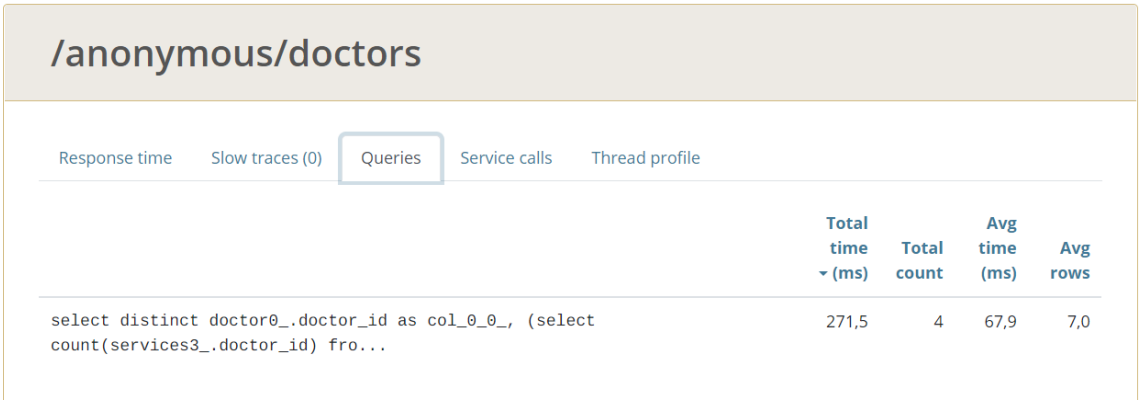
- También hemos cacheado este mismo servicio:

```java
@Transactional(readOnly = true)
@Cacheable("doctorsWithServices")
public Collection<Doctor> findAllDoctorsWithServices() throws DataAccessException {
    return this.doctorRepository.findDoctorsWithServices();
}
```

Y lo hemos añadido al respectivo archivo xml ehcache3:

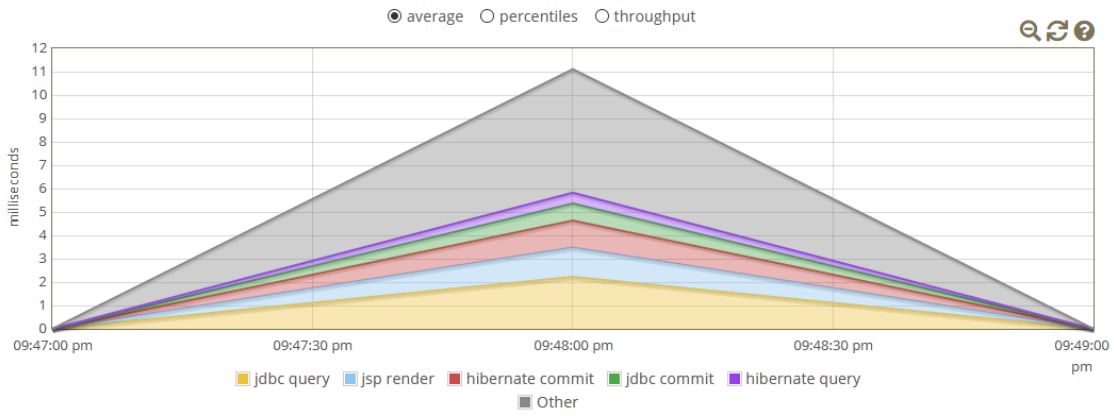
```
    </cache>
    <cache alias="doctorsWithServices" uses-template="default">
        <key-type>org.springframework.cache.interceptor.SimpleKey</key-type>
        <value-type>java.util.Collection</value-type>
    </cache>
</config>
```

En la siguiente captura, observamos los tiempos de mejora de la consulta. Además, podemos observar que una de las queries, la asociada a la consulta de los servicios del médico ha desaparecido, ya que, en la consulta de este, nos estamos trayendo también los servicios que ofrece, mejorando así la optimización del tiempo empleado:



- HU-005 y HU-006: Cambio de médico de cabecera – Restricción del cambio médico de cabecera

All Web Transactions



Queries /patient/edit

Response time | Slow traces (0) | Queries | Service calls | Thread profile

	Total time (ms)	Total count	Avg time (ms)	Avg rows
select patient0_.patient_id as id1_17_, patient0_1_.dni as dni2_17_, patient0_1_.email...	65,9	160	0.41	0,8
select doctor0_.doctor_id as id1_17_, doctor0_1_.dni as dni2_17_, doctor0_1_.email as ...	39,9	80	0.50	5,0
select services0_.doctor_id as doctor_i1_10_0_, services0_.service_id as service_2_10_...	21,4	40	0.54	0
select patient0_.patient_id as id1_17_, patient0_1_.dni as dni2_17_, patient0_1_.email...	19,5	40	0.49	4,0
select doctor0_.doctor_id as id1_17_, doctor0_1_.dni as dni2_17_, doctor0_1_.email as ...	16,5	40	0.41	1,0
update patients set address=?, birth_date=?, general_practitioner_id=?, nss=?, phone2=...	14,7	40	0.37	1,0

Hemos hecho los cambios convenientes en los servicios, repositorios y controladores, así como en el XML de elementos cachables.

En el XML mencionado, se han añadido las siguientes líneas:

```
46        <cache alias="services" uses-template="default">
47            <key-type>org.springframework.cache.interceptor.SimpleKey</key-type>
48            <value-type>java.util.Collection</value-type>
49        </cache>
50
51        <cache alias="doctors" uses-template="default">
52            <key-type>org.springframework.cache.interceptor.SimpleKey</key-type>
53            <value-type>java.util.Collection</value-type>
54        </cache>
55
56        <cache alias="doctorsS" uses-template="default">
57            <key-type>org.springframework.cache.interceptor.SimpleKey</key-type>
58            <value-type>java.util.Collection</value-type>
59        </cache>
```

Hemos modificado la clase DoctorRepository, con su correspondiente query.

```
12    Collection<Doctor> findAll() throws DataAccessException;
13
14    Collection<Doctor> findAllDoctorsAndServices() throws DataAccessException;
15
16    Collection<Doctor> findDoctorsSortedByNumOfServices() throws DataAccessException;
17
```

```
49    Collection<Service> findAllServices();
50
51    @Override
52    @Query("SELECT doctor from Doctor doctor left join fetch doctor.services services")
53    Collection<Doctor> findAllDoctorsAndServices();
54
55 }
```

Hemos añadido las anotaciones necesarias en los métodos dentro de la clase DoctorService.

```
38    @Transactional(readOnly = true)
39    @Cacheable("doctors")
40    public Collection<Doctor> findAllDoctors() throws DataAccessException {
41        return this.doctorRepository.findAll();
42    }
43
44
45    @Transactional(readOnly = true)
46    @Cacheable("doctorsS")
47    public Collection<Doctor> findAllDoctorsAndServices() throws DataAccessException {
48        return this.doctorRepository.findAllDoctorsAndServices();
49    }
```

```
79    @Cacheable("services")
80    @Transactional(readOnly = true)
81    public Collection<org.springframework.clinicaetsii.model.Service> findAllServices() {
82        return this.doctorRepository.findAllServices();
83    }
84
```

En la clase PatientService, hemos añadido la anotación indicada para vaciar la caché.

```
119
120   @Transactional
121   @CacheEvict(cacheNames = "doctorsS", allEntries = true)
122   public void save(final Patient patient) throws DataAccessException {
123       this.patientRepository.save(patient);
124   }
125
```

Y en la clase PatientPatientController, hemos modificado lo siguiente:

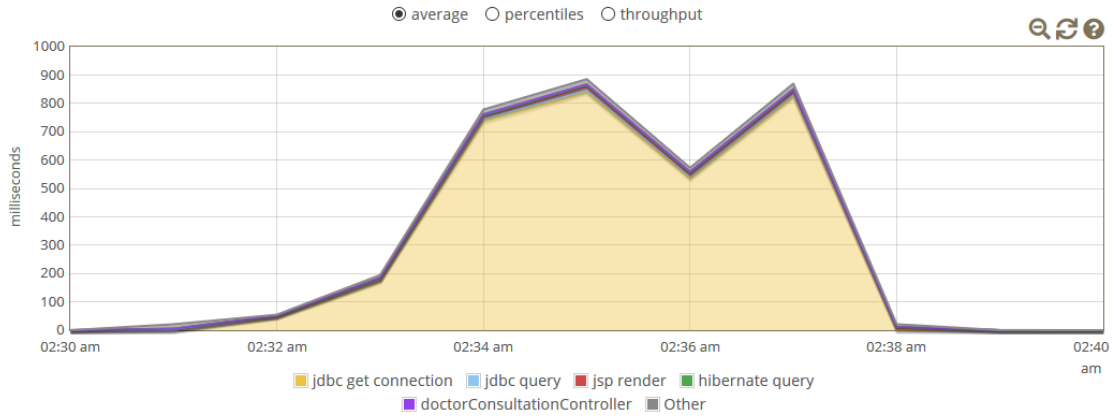
```
52
53    @ModelAttribute("doctors")
54    public Collection<Doctor> populateDoctors() {
55        return this.doctorService.findAllDoctorsAndServices();
56    }
57
58    @GetMapping("/patient")
```

Aunque las consultas no sean muy costosas, cachearlas es una buena práctica, ya que son frecuentes y el resultado de éstas no varía.

	(ms)	count	(ms)	rows
select patient0_.patient_id as id1_17_, patient0_1_.dni as dni2_17_, patient0_1_.email...	55,6	160	0.35	0,8
select doctor0_.doctor_id as id1_17_0_, service2_.id as id1_16_1_, doctor0_1_.dni as d...	39,0	80	0.49	7,0
select patient0_.patient_id as id1_17_, patient0_1_.dni as dni2_17_, patient0_1_.email...	35,4	40	0.89	4,0
update patients set address=?, birth_date=?, general_practitioner_id=?, nss=?, phone2=...	14,8	40	0.37	1,0
select doctor0_.doctor_id as id1_17_, doctor0_1_.dni as dni2_17_, doctor0_1_.email as ...	14,7	40	0.37	1,0

- HU-016: Creación de una nueva consulta a partir de una cita

All Web Transactions



Queries /doctor/appointments

	Total time (ms)	Total count	Avg time (ms)	Avg rows
select appointmen0_.id as id1_1_, appointmen0_.end_time as end_time2_1_, appointmen...	11.990,5	8,192	1,5	0,7
select patient0_.patient_id as id1_17_0_, patient0_1_.dni as dni2_17_0_, patient0_1...	7060,2	5,594	1,3	1,0

```java
package org.springframework.clinicaetsii.model.projection;

import java.time.LocalDateTime;

public interface AppointmentPatient {

    Integer getPatientId();

    Integer getAppointmentId();

    String getName();

    String getSurname();

    LocalDateTime getStartTime();

    LocalDateTime getEndTime();
}
```

```
@Override
@Query("SELECT DISTINCT appointment.patient.id AS patientId, appointment.patient.name AS name, appointment.patient.surname AS surname, "
        + "appointment.id AS appointmentId, appointment.startTime AS startTime, "
        + "appointment.endTime AS endTime FROM Appointment appointment "
        + "WHERE (appointment.patient.generalPractitioner.username LIKE :doctorUsername) AND NOT EXISTS (SELECT consultation FROM Consult
Collection<AppointmentPatient> findAppointmentsPatientsWithoutConsultationByDoctorUsername(
        @Param("doctorUsername") String doctorUsername);
```

Hemos hecho los cambios covenientes en el servicio, controlador y vista.

```
@Data
@EqualsAndHashCode(callSuper = false)
@Entity
@Table(name = "appointments", indexes = {@Index(columnList = "priority, start_time desc")})
public class Appointment extends BaseEntity {
```

Hemos añadido las columnas utilizadas en la condición de la query como índice de la tabla para acelerar la consulta.

/doctor/patients/*/consultations/new

	Total time (ms)	Total count	Avg time (ms)	Avg rows
insert into consultations (anamnesis, appointment_id, discharge_type_id, end_time, ...	16.076,0	8,192	2,0	1,0
select appointmen0_.id as id1_1_0_, appointmen0_.end_time as end_time2_1_0_, appoin...	16.851,1	16,384	1,0	1,0
select services0_.doctor_id as doctor_i1_10_0_, services0_.service_id as service_2_...	12.763,7	16,384	0.78	1,0
select diagnosis0_.id as id1_8_, diagnosis0_.name as name2_8_ from diagnoses diagno...	9766,4	16,384	0.60	3,0
select discharget0_.id as id1_9_, discharget0_.name as name2_9_ from discharge_type...	9269,7	16,384	0.57	6,0

Cambiamos el FetchType para que cada vez que hagamos esas consultas no se traiga todos los datos si no son necesarios.

```
@ManyToMany(fetch = FetchType.LAZY)
@JoinTable(name = "doctor_services", joinColumns = @JoinColumn(name = "doctor_id"),
        inverseJoinColumns = @JoinColumn(name = "service_id"))
private Collection<Service> services;
```

```
package org.springframework.clinicaetsii.configuration;

import org.springframework.cache.annotation.EnableCaching;
import org.springframework.context.annotation.Configuration;

@Configuration
@EnableCaching
public class CacheConfiguration {

}
```

```java
package org.springframework.clinicaetsii.configuration;

import org.ehcache.event.CacheEvent;
import org.ehcache.event.CacheEventListener;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class CacheLogger implements CacheEventListener<Object, Object> {

    private final Logger LOG = LoggerFactory.getLogger(CacheLogger.class);

    @Override
    public void onEvent(final CacheEvent<? extends Object, ? extends Object> event) {
        this.LOG.info("Key: {} | EventType: {} | Old value {} | New value: {}", event.getKey(),
                event.getType(), event.getOldValue(), event.getNewValue());
    }
}
```

Hemos hecho los cambios convenientes en los servicios, repositorios y controladores, así como en el XML de elementos cacheables.

En el XML mencionado, se han añadido las siguientes líneas:

```xml
    <cache alias="diagnoses" uses-template="default">
        <key-type>org.springframework.cache.interceptor.SimpleKey</key-type>
        <value-type>java.util.Collection</value-type>
    </cache>

    <cache alias="dischargeTypes" uses-template="default">
        <key-type>org.springframework.cache.interceptor.SimpleKey</key-type>
        <value-type>java.util.Collection</value-type>
    </cache>

    <cache alias="constantTypes" uses-template="default">
        <key-type>org.springframework.cache.interceptor.SimpleKey</key-type>
        <value-type>java.util.Collection</value-type>
    </cache>
```

Hemos añadido las anotaciones necesarias en los métodos dentro de la clase DoctorService.

```java
@Transactional(readOnly = true)
@Cacheable("dischargeTypes")
public Collection<DischargeType> findDischargeTypes() throws DataAccessException {
    return this.consultationRepository.findDischargeTypes();
}

@Transactional(readOnly = true)
@Cacheable("diagnoses")
public Collection<Diagnosis> findAllDiagnoses() throws DataAccessException {
    return this.diagnosisRepository.findAll();
}
```

Hemos añadido las anotaciones necesarias en los métodos dentro de la clase ConstantService.

```java
@Transactional(readOnly = true)
@Cacheable("constantTypes")
public Collection<ConstantType> findAllConstantTypes() {
    return this.constantRepository.findAllConstantTypes();
}
```

Hemos añadido una cache y tres funciones de los servicios.

Aunque las consultas no sean muy costosas es bueno cachearlas pues son muy repetitivas y el resultado es siempre el mismo, a parte se utilizan mucho.

/doctor/patients/*/consultations/*

	Total time (ms)	Total count	Avg time (ms)	Avg rows
select examinatio1_.id as id1_12_, examinatio1_.description as descript2_12_, exa...	88.781,1	20,463	4,3	1784,9
select consultati0_.id as id1_7_0_, consultati0_.anamnesis as anamnesi2_7_0_, con...	20.495,6	20,463	1,0	1,0
select constants0_.consultation_id as consulta1_5_0_, constants0_.constant_id as ...	19.788,5	20,463	0.97	0,3
select diagnoses0_.consultation_id as consulta1_6_0_, diagnoses0_.diagnosis_id as...	19.783,1	20,463	0.97	0,6
select doctor0_.doctor_id as id1_17_0_, doctor0_1_.dni as dni2_17_0_, doctor0_1_....	19.078,6	20,463	0.93	1,0
select diagnosis0_.id as id1_8_, diagnosis0_.name as name2_8_ from diagnoses diag...	12.009,3	20,463	0.59	3,0
select discharget0_.id as id1_9_, discharget0_.name as name2_9_ from discharge_ty...	10.623,7	20,463	0.52	6,0

Para este tipo de consulta también se ha utilizado la implementación de la caché antes mencionada.

/doctor/patients/*/consultations/*/edit

	Total time (ms)	Total count	Avg time (ms)	Avg rows
update consultations set anamnesis=?, appointment_id=?, discharge_type_id=?, end_...	68.456,9	8,168	8,4	1,0
select examinatio0_.consultation_id as consulta4_12_0_, examinatio0_.id as id1_12...	61.449,6	8,192	7,5	3070,2
delete from consultation_diagnoses where consultation_id=?	13.071,0	8,174	1,6	1,0
insert into consultation_diagnoses (consultation_id, diagnosis_id) values (?, ?)	9658,1	8,174	1,2	1,0
select consultati0_.id as id1_7_0_, consultati0_.anamnesis as anamnesi2_7_0_, con...	15.252,5	16,384	0.93	1,0
select appointmen0_.id as id1_1_0_, appointmen0_.end_time as end_time2_1_0_, appo...	7557,8	8,192	0.92	1,0
select diagnoses0_.consultation_id as consulta1_6_0_, diagnoses0_.diagnosis_id as...	14.564,4	16,368	0.89	0,9
select doctor0_.doctor_id as id1_17_0_, doctor0_1_.dni as dni2_17_0_, doctor0_1_....	7214,3	8,192	0.88	1,0
select services0_.doctor_id as doctor_i1_10_0_, services0_.service_id as service_...	5735,9	8,192	0.70	1,0
select diagnosis0_.id as id1_8_, diagnosis0_.name as name2_8_ from diagnoses diag...	13.077,6	24,576	0.53	3,0
select discharget0_.id as id1_9_, discharget0_.name as name2_9_ from discharge_ty...	11.747,2	24,576	0.48	6,0

Para este tipo de consulta también se ha utilizado la implementación de la caché antes mencionada.

```
@Override
@Query("SELECT c FROM Consultation c LEFT JOIN FETCH c.examinations examinations "
        + "WHERE c.id = :id")
Consultation findFullConsultation(@Param("id") int id);
```

/doctor/patients/*/consultations/*/constants/new

	Total time (ms)	Total count	Avg time (ms)	Avg rows
insert into constants (constant_type_id, value_constant) values (?, ?)	2,0	1	2,0	1,0
insert into consultation_constants (consultation_id, constant_id) values (?, ?)	1,3	1	1,3	1,0
select consultati0_.id as id1_7_0_, consultati0_.anamnesis as anamnesi2_7_0_, consu...	3750,4	4,096	0.92	1,0
select doctor0_.doctor_id as id1_17_0_, doctor0_1_.dni as dni2_17_0_, doctor0_1_.em...	3627,9	4,096	0.89	1,0
select constants0_.consultation_id as consulta1_5_0_, constants0_.constant_id as co...	3098,4	4,096	0.76	1,0
select constantty0_.id as id1_3_, constantty0_.name as name2_3_ from constant_types...	7088,6	12,288	0.58	19,0

Para este tipo de consulta también se ha utilizado la implementación de la caché antes mencionada.

/doctor/patients/*/consultations/*/examinations/new

	Total time (ms)	Total count	Avg time (ms)	Avg rows
update examinations set consultation_id=? where id=?	42.294,0	4,096	10,3	1,0
select examinatio0_.consultation_id as consulta4_12_0_, examinatio0_.id as id1_12...	25.623,5	4,096	6,3	2046,0
insert into examinations (description, start_time) values (?, ?)	8044,9	4,096	2,0	1,0
select consultati0_.id as id1_7_0_, consultati0_.anamnesis as anamnesi2_7_0_, con...	9171,4	8,192	1,1	1,0
select doctor0_.doctor_id as id1_17_0_, doctor0_1_.dni as dni2_17_0_, doctor0_1_....	8352,5	8,192	1,0	1,0

```
@Override
@Query("SELECT c FROM Consultation c LEFT JOIN FETCH c.examinations examinations "
        + "WHERE c.id = :id")
Consultation findFullConsultation(@Param("id") int id);
```

Cambiamos el FetchType para que cada vez que hagamos esas consultas no se traiga todos los datos si no son necesarios.

```
@OneToMany(fetch = FetchType.LAZY)
@JoinColumn(name = "consultation_id")
private Collection<Examination> examinations;

@ManyToMany(fetch = FetchType.LAZY)
@JoinTable(name = "consultation_diagnoses", joinColumns = @JoinColumn(name = "consultation_id"),
        inverseJoinColumns = @JoinColumn(name = "diagnosis_id"))
private Collection<Diagnosis> diagnoses;

@ManyToMany(fetch = FetchType.LAZY)
@JoinTable(name = "consultation_constants", joinColumns = @JoinColumn(name = "consultation_id"),
        inverseJoinColumns = @JoinColumn(name = "constant_id"))
private Collection<Constant> constants;
```

Para evitar consultar examinations, diagnoses y constants siempre que no sea necesario.