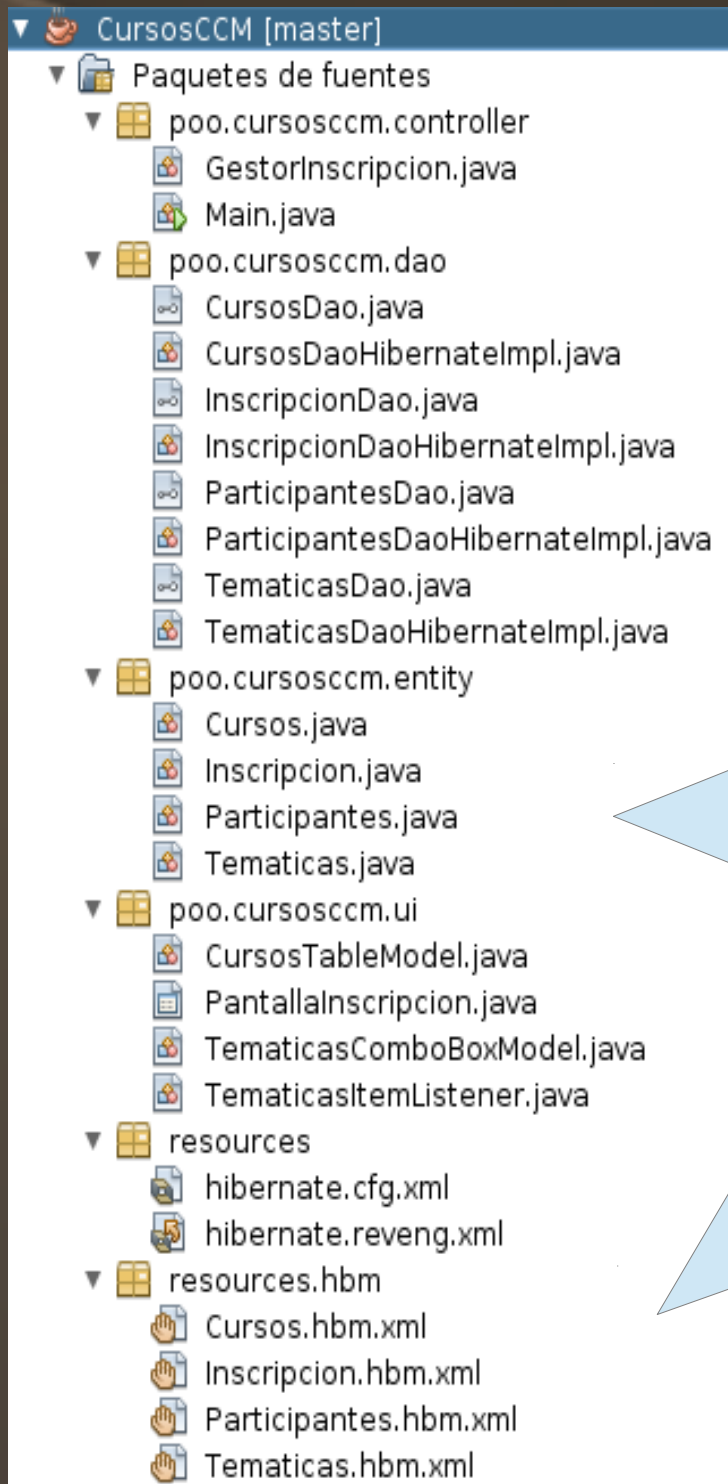


Implementado con

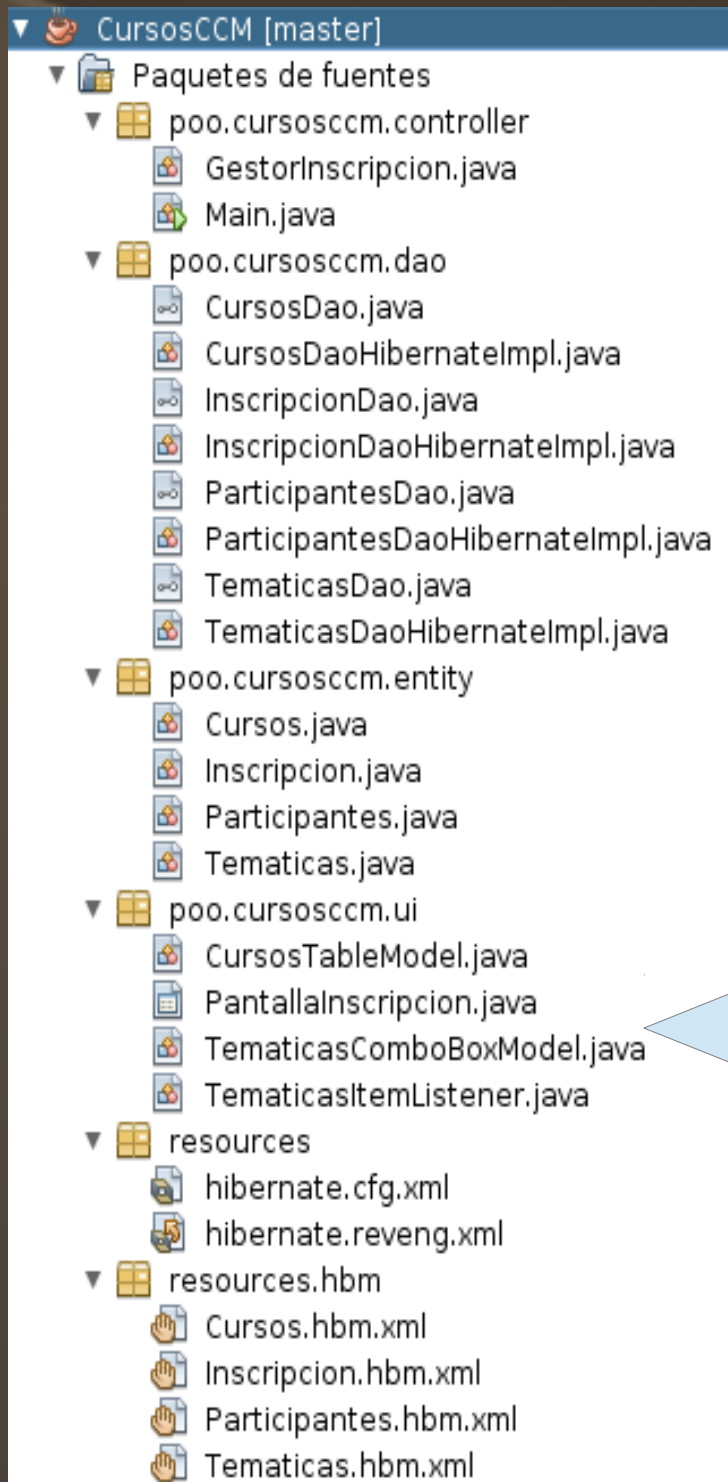
Patrón MVC + Patrón DAO



Patrón MVC

Modelo

mapeo



Patrón MVC

Vista

Paquete poo.cursosccm.ui (vista)

Formulario de Inscripcion a Cursos del Centro Cultural Municipal

Estudiante

☒ Estudiante Registrado Ingrese su e-mail:

☐ Estudiante No Registrado Nombre: ---
Apellido: ---

Cursos Vigentes

Temáticas:

Nombre	Profesor	Aula	Duración	Carga Horaria	Cupo
--------	----------	------	----------	---------------	------

Paquete poo.cursosccm.ui (vista)

```
public class TematicasComboBoxModel extends AbstractListModel implements ComboBoxModel{

    private final List<Tematicas> tematicas;
    private Tematicas tematicaSeleccionada;
    private final static int FIRSTINDEX = 0;

    public TematicasComboBoxModel(){...3 lines }

    public TematicasComboBoxModel(List<Tematicas> tematicas) {...7 lines }

    @Override
    public void setSelectedItem(Object anItem) {...3 lines }

    @Override
    public Object getSelectedItem() {...3 lines }

    @Override
    public int getSize() {...3 lines }

    @Override
    public Object getElementAt(int index) {...3 lines }

    public Tematicas obtenerTematicaEn(int indice){...3 lines }

}
```

Paquete poo.cursosccm.ui (vista)

```
public class TematicasItemListener implements ItemListener{

    private final PantallaInscripcion pantallaInscripcion;
    private final GestorInscripcion gestor;
    private CursosTableModel cursosTableModel;

    public TematicasItemListener(PantallaInscripcion pantallaInscripcion, GestorInscripcion gestor) {
        this.gestor = gestor;
        this.pantallaInscripcion = pantallaInscripcion;
    }

    @Override
    public void itemStateChanged(ItemEvent e) {
        // obtenemos el la tematica seleccionado
        Tematicas tematicaSeleccionada = pantallaInscripcion.obtenerTematicaSeleccionada();
        //si el item no es nulo
        if (tematicaSeleccionada != null) {
            //rellenamos el modelo de la tabla
            cursosTableModel = new CursosTableModel(gestor.listarCursos(tematicaSeleccionada));
            //asignamos el modelo a la tabla
            pantallaInscripcion.setTableModel(cursosTableModel);
        }
    }
}
```

Paquete poo.cursosccm.ui (vista)

```
public class CursosTableModel extends AbstractTableModel{

    private static final String[] COLUMNAS = { "Nombre", "Profesor", "Aula", "Duración", "Carga Horaria", "Cupo" };

    private List<Cursos> cursos;

    public CursosTableModel() {...2 lines }

    public CursosTableModel(List<Cursos> cursos) {...4 lines }

    @Override
    public int getRowCount() {...3 lines }

    @Override
    public int getColumnCount() {...3 lines }

    @Override
    public Object getValueAt(int fila, int columna) {...25 lines }

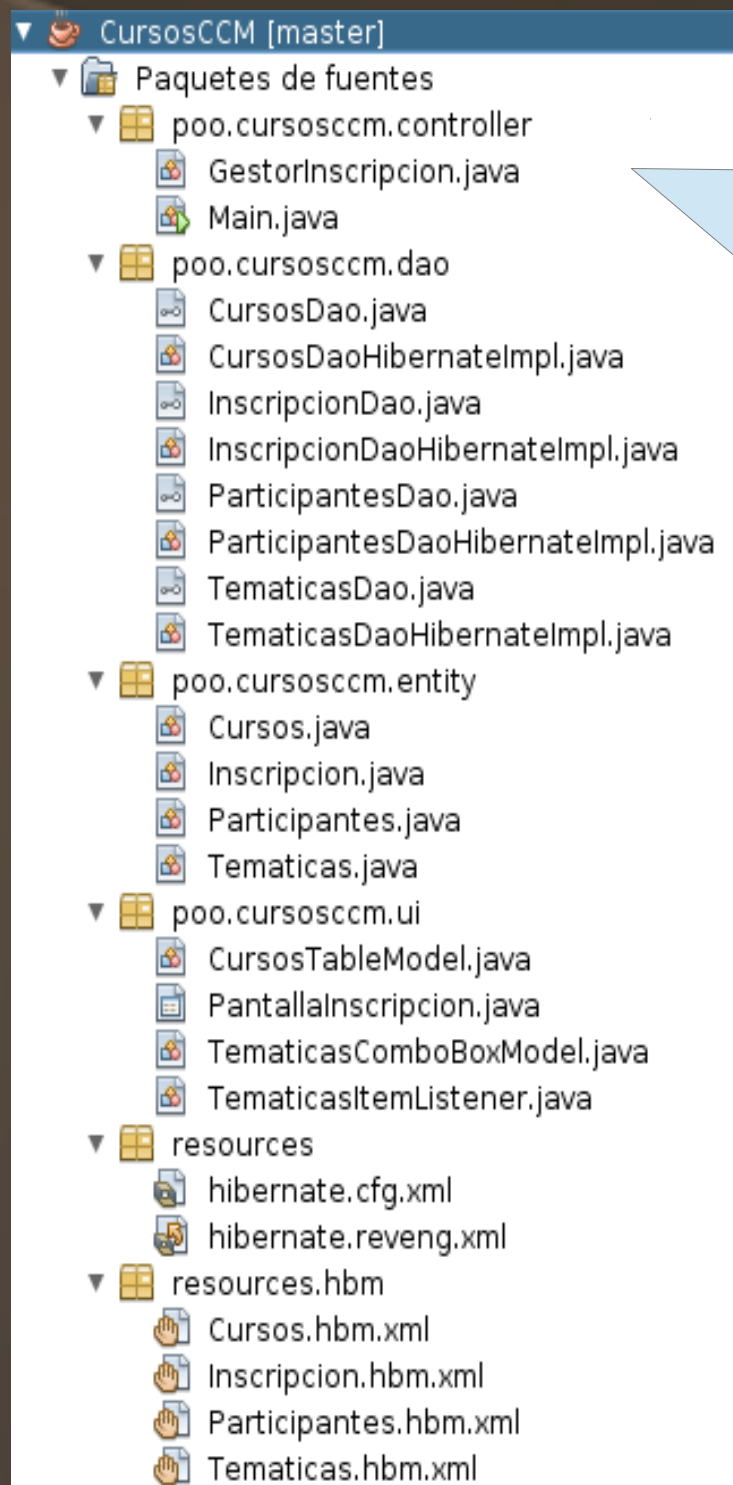
    @Override
    public String getColumnName(int index) {
        return COLUMNAS[index];
    }

    public Cursos obtenerCursoEn(int fila) {...3 lines }

    public void setCursos(List<Cursos> cursos) {...3 lines }

    public List<Cursos> getCursos(){...3 lines }

}
```

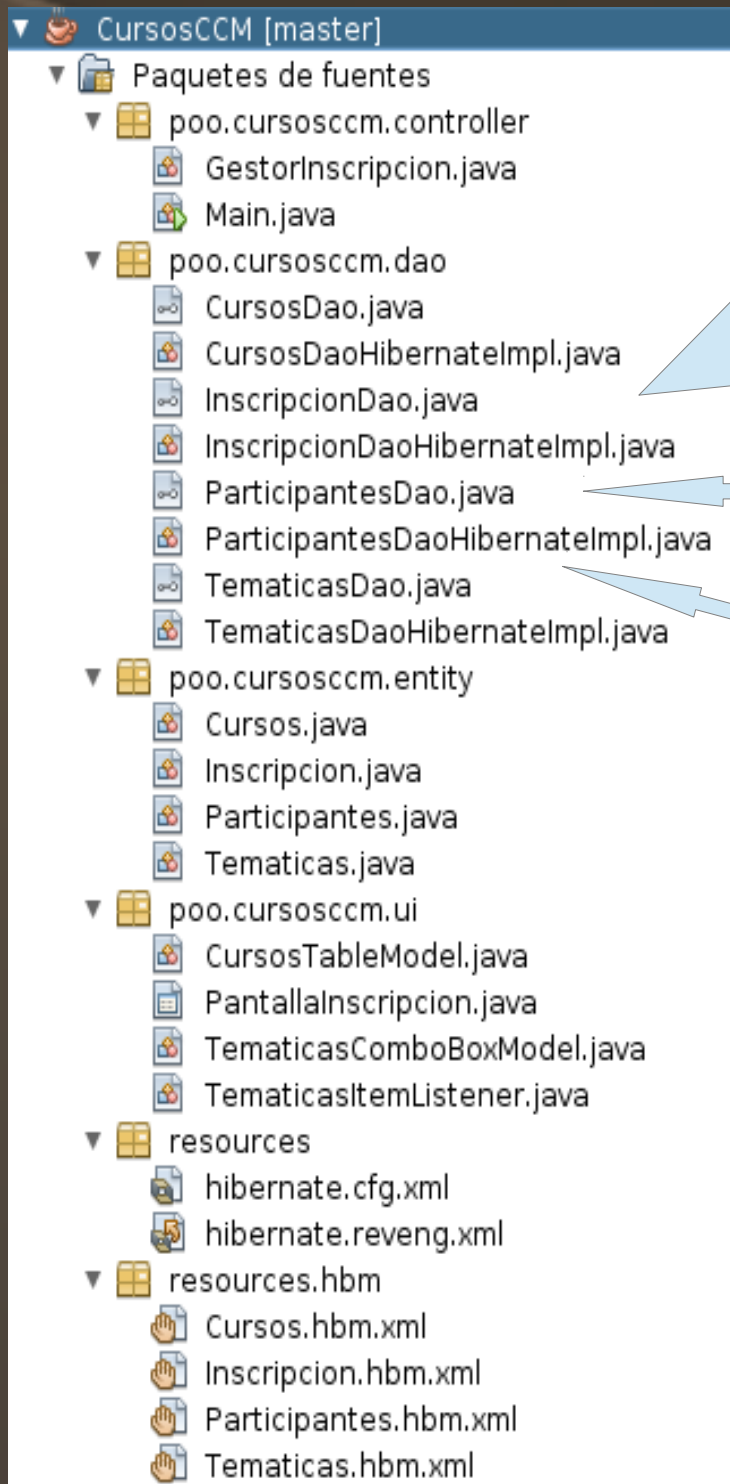


Patrón MVC

Controlador

Paquete poo.cursosccm.controller

```
30 public class GestorInscripcion {
31
32     private final CursosDao cursosDao;
33     private final TematicasDao tematicasDao;
34     private final ParticipantesDao participantesDao;
35     private final InscripcionDao inscripcionDao;
36
37     public GestorInscripcion(SessionFactory sessionFactory) {
38         // creamos las instancias de los objetos de acceso a datos
39         this.cursosDao = new CursosDaoHibernateImpl(sessionFactory);
40         this.tematicasDao = new TematicasDaoHibernateImpl(sessionFactory);
41         this.participantesDao = new ParticipantesDaoHibernateImpl(sessionFactory);
42         this.inscripcionDao = new InscripcionDaoHibernateImpl(sessionFactory);
43     }
44
45     /** ...3 lines */
48     public void run() {...3 lines }
51
52     public List<Cursos> listarCursos(Tematicas tematica){...3 lines }
55
56     public List<Tematicas> listarTematicas(){...3 lines }
59
60     public Participantes buscarParticipante(String mail){...3 lines }
63
64     public void generarInscripcion(Participantes participante, Cursos curso){...44 lines }
108
109 }
```



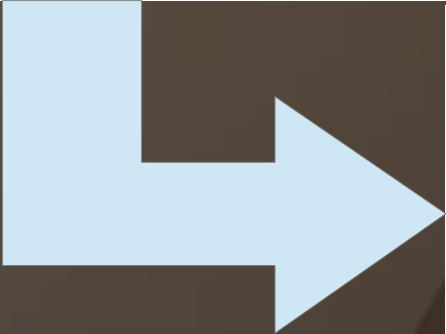
Patrón DAO

Interfaces

**Implementaciones
En
Hibernate**

Paquete poo.cursosccm.dao

```
public interface ParticipantesDao {  
  
    /**  
     * Decuelve una única instancia de Participantes que coincide  
     * con el campo email que recibe como parámetro.  
     * @param email  
     * @return Participantes  
     */  
    public Participantes buscarParticipanteXemail(String email);  
}
```



```
public class ParticipantesDaoHibernateImpl implements ParticipantesDao{  
  
    private SessionFactory sessionFactory;  
    private Participantes participante;  
  
    public ParticipantesDaoHibernateImpl(SessionFactory sessionFactory) {  
        this.sessionFactory = sessionFactory;  
    }  
  
    @Override  
    public Participantes buscarParticipanteXemail(String email) {  
        Session session = sessionFactory.openSession();  
        CriteriaBuilder builder = session.getCriteriaBuilder();  
        CriteriaQuery<Participantes> query = builder.createQuery(Participantes.class);  
        Root<Participantes> root = query.from(Participantes.class);  
        query.select(root);  
        query.where(builder.equal(root.get("email"), email));  
        participante = session.createQuery(query).uniqueResult();  
        session.close();  
        return participante;  
    }  
}
```

Paquete poo.cursosccm.dao

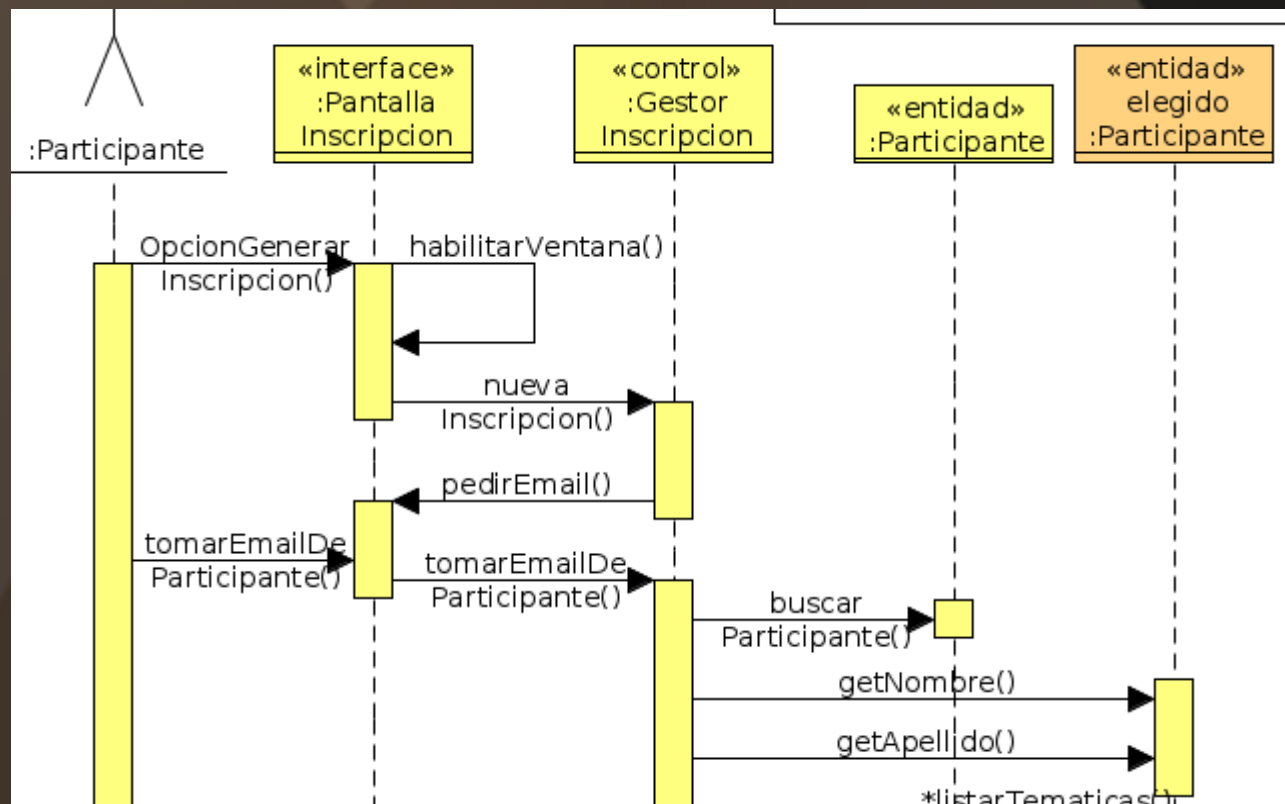
```
17 public interface IncripcionDao {
18     /**
19      * Devuelve un entero que representa la cantidad de cursos en los que
20      * el participante, pasado como parámetro, está inscripto.
21      * @param participante
22      * @return int: que representa la cantidad de cursos en los que el participante está inscripto
23      */
24     public int getCantidadCursosPorParticipante(Participantes participante);
25
26     /**
27      * Devuelve un booleano que indica si el participante pasado como parámetro
28      * se encuentra inscripto en ese curso.
29      * Esto sirve para evitar que un participante se inscriba dos veces en el
30      * mismo curso.
31      * @param participante
32      * @param curso
33      * @return true: si el participante se encuentra inscripto en ese curso
34      *         false: en caso contrario
35      */
36     public boolean estaInscripto(Participantes participante, Cursos curso);
37
38     /**
39      * Devuelve un booleano que indica si quedan vacantes en el cupo
40      * del curso pasado como parámetro.
41      * @param curso
42      * @return true: si hay cupos disponibles
43      *         false: en caso contrario
44      */
45     public boolean hayVacantes(Cursos curso);
46
47     /**
48      * Guarda la inscripcion en la base de datos
```

Paquete poo.cursosccm.dao

```
20 public class IncripcionDaoHibernateImpl implements IncripcionDao{
21
22     private final SessionFactory sessionFactory;
23
24     public IncripcionDaoHibernateImpl(SessionFactory sessionFactory) {...3 lines }
25
26     @Override
27     public int getCantidadCursosPorParticipante(Participantes participante) {...10 lines }
28
29     @Override
30     public boolean estaInscripto(Participantes participante, Cursos curso) {...15 lines }
31
32     @Override
33     public boolean hayVacantes(Cursos curso) {
34         int cantidad;
35         try (Session session = sessionFactory.openSession()) {
36             String consulta = "select count(i.idinscripcion) as cantidad from Incripcion i where i.cursos = :c";
37             Query query = session.createQuery(consulta);
38             query.setParameter("c", curso);
39             cantidad = Integer.parseInt(query.getSingleResult().toString());
40         }
41         return cantidad < curso.getCupo();
42     }
43
44     @Override
45     public int guardar(Incripcion inscripcion) {...9 lines }
46
47 }
```

1. **P:** selecciona la opción **Inscribirse a curso**.
2. **Sistema:** solicita se confirme si se trata de un participante registrado.
3. **P:** confirma que está registrado.
4. **Sistema:** solicita se ingrese e-mail del participante.
5. **P:** ingresa e-mail
6. **Sistema:** busca participante con ese mail y lo encuentra, mostrando nombre y apellido.

Secuencia buscar Participante



Ingrese su e-mail:

Buscar

Nombre: ---

Apellido: ---

```
private void btnBuscarActionPerformed(java.awt.event.ActionEvent evt) {  
    //verificamos si el participante existe en la base de datos  
    participante = gestor.buscarParticipante(txtEmail.getText());  
    if(participante!=null){  
        //si existe habilitamos la inscripcion  
        habilitarInscripcion();  
    }else{  
        // si no existe deshabilitamos la inscripcion  
        deshabilitarInscripcion();  
    }  
}
```

```
public Participantes buscarParticipante(String mail){  
    return participantesDao.buscarParticipanteXemail(mail);  
}
```

```
public class ParticipantesDaoHibernateImpl implements ParticipantesDao{  
  
    private SessionFactory sessionFactory;  
    private Participantes participante;  
  
    public ParticipantesDaoHibernateImpl(SessionFactory sessionFactory) {  
        this.sessionFactory = sessionFactory;  
    }  
  
    @Override  
    public Participantes buscarParticipanteXemail(String email) {  
        Session session = sessionFactory.openSession();  
        CriteriaBuilder builder = session.getCriteriaBuilder();  
        CriteriaQuery<Participantes> query = builder.createQuery(Participantes.class);  
        Root<Participantes> root = query.from(Participantes.class);  
        query.select(root);  
        query.where(builder.equal(root.get("email"), email));  
        participante = session.createQuery(query).uniqueResult();  
        session.close();  
        return participante;  
    }  
}
```

7. **Sistema:** busca y muestra las temáticas de los cursos y solicita se seleccione una.

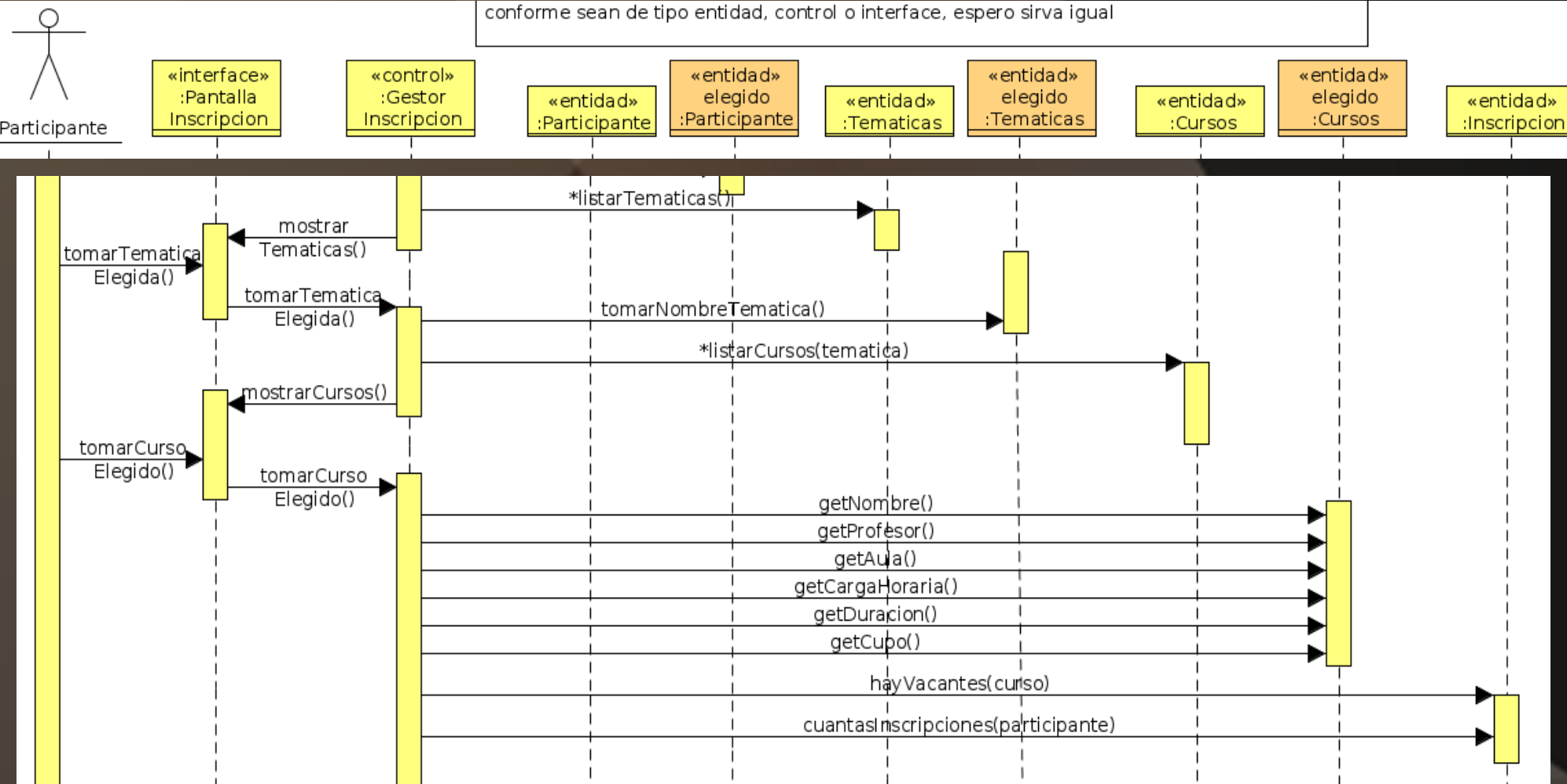
8. **P:** selecciona temática del curso.

8. **Sistema:** para la temática seleccionada muestra los cursos vigentes junto con los siguientes datos: duración, nombre, carga horaria, nro. de aula, nombre y apellido del profesor y cupo, y solicita se seleccione uno.

9. **P:** selecciona un curso

Secuencia habilitar Inscripción

conforme sean de tipo entidad, control o interface, espero sirva igual




```

323 private void habilitarInscripcion() {
324     cbTematicas.setModel(tematicasComboBoxModel);
325     cbTematicas.addItemListener(tematicasItemListener);
326     cursosTableModel = new CursosTableModel(gestor.listarCursos(this.obtenerTematicaSeleccionada()));
327     tablaCursos.setModel(cursosTableModel);
328     rbEstudianteRegistrado.setSelected(true);
329     lblApellido.setText(gestor.buscarParticipante(txtEmail.getText()).getApellido());
330     lblNombre.setText(gestor.buscarParticipante(txtEmail.getText()).getNombre());
331     cbTematicas.setEnabled(true);
332     tablaCursos.setEnabled(true);
333     tablaCursos.setVisible(true);
334 }

```

Estudiante

☒ Estudiante Registrado

Ingrese su e-mail:

juanperez@gmail.com

Buscar

☐ Estudiante No Registrado

Nombre: Juan

Apellido: Perez

Cursos Vigentes

Temáticas:

Literatura

Nombre	Profesor	Aula	Duración	Carga Horaria	Cupo
Literatura Moderna	Haruki Murakami	23	2 meses	6 hs/semana	10
Taller Literario	Paul Auster	1	6 meses	2 hs/semana	2

```

@Override
public void itemStateChanged(ItemEvent e) {
    // obtenemos el la tematica seleccionado
    Tematicas tematicaSeleccionada = pantallaInscripcion.obtenerTematicaSeleccionada();
    //si el item no es nulo
    if (tematicaSeleccionada != null) {
        //rellenamos el modelo de la tabla
        cursosTableModel = new CursosTableModel(gestor.listarCursos(tematicaSeleccionada));
        //asignamos el modelo a la tabla
        pantallaInscripcion.setTableModel(cursosTableModel);
    }
}

```

```

52 public List<Cursos> listarCursos(Tematicas tematica){
53     return cursosDao.listarCursos(tematica);
54 }

```

```

18 public class CursosDaoHibernateImpl implements CursosDao {
19
20     private final SessionFactory sessionFactory;
21
22     public CursosDaoHibernateImpl(SessionFactory sessionFactory) {...3 lines }
23
24
25
26     @Override
27     public List<Cursos> listarCursos(Tematicas tematica) {
28         Session session = sessionFactory.openSession();
29         CriteriaBuilder builder = session.getCriteriaBuilder();
30         CriteriaQuery<Cursos> query = builder.createQuery(Cursos.class);
31         Root<Cursos> root = query.from(Cursos.class);
32         query.select(root);
33         query.where(builder.equal(root.get("tematicas"), tematica));
34         List<Cursos> cursos = session.createQuery(query).list();
35         session.close();
36         return cursos;
37     }
}

```

10. Sistema: valida que el participante no curse simultáneamente más de dos cursos, y no los cursa. (ver RN 1: Cursos inscriptos por participantes)

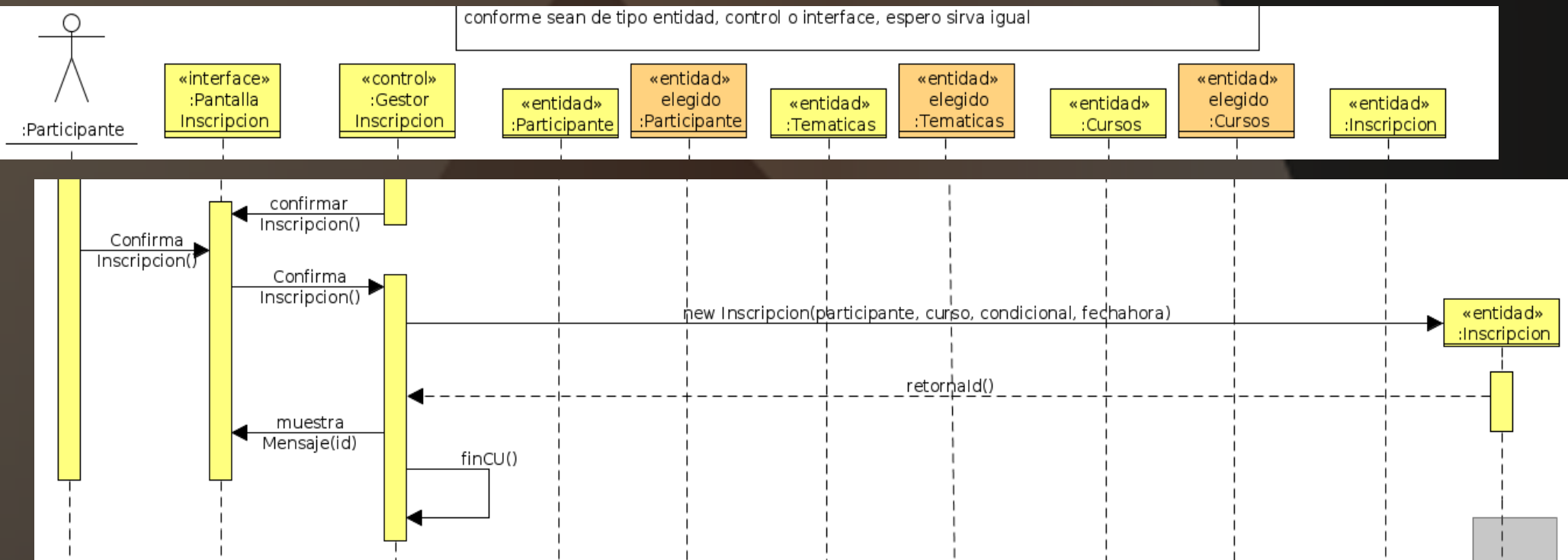
11. SI: solicita se confirme la inscripción al curso.

12. P: confirma la inscripción al curso.

13. Sistema: registra la inscripción al curso seleccionado con los siguientes datos: nro. de inscripción, fecha y hora de inscripción, nombre del curso, nombre, apellido y mail del participante.

14. Sistema: valida que el curso no esté completo y no lo está. Actualiza el estado de la inscripción a Inscripto y lo informa. Fin del escenario del caso de uso

Secuencia confirmar inscripción



Formulario de Inscripcion a Cursos del Centro Cultural Municipal

Estudiante

☒ Estudiante Registrado Ingrese su e-mail:

☐ Estudiante No Registrado Nombre: Juan
 Apellido: Perez

Cursos Vigentes

Temáticas:

Nombre	Profesor	Aula	Duración	Carga Horaria	Cupo
Literatura Moderna	Haruki Murakami	23	2 meses	6 hs/semana	10
Taller Literario	Paul Auster	1	6 meses	2 hs/semana	2

```

267 private void btnInscribirseActionPerformed(java.awt.event.ActionEvent evt) {
268     obtenerCursoSeleccionado();
269     //si se ha seleccionado un curso de la tabla generamos la inscripción
270     if(cursoSeleccionado!=null){
271         gestor.generarInscripcion(participante, cursoSeleccionado);
272     }else{
273         JOptionPane.showMessageDialog(null, "No ha seleccionado ningún curso.\n"
274             + "Debe seleccionar un curso para inscribirse", "Advertencia", JOptionPane.WARNING_MESSAGE);
275     }
276 }
  
```

```

64 public void generarInscripcion(Participantes participante, Cursos curso){
65     Calendar ahora = Calendar.getInstance();
66     ahora.add(Calendar.MINUTE, -55);
67     //validamos que no exista una inscripcion de ese participante para ese curso
68     //esta no es una regla de negocio explicita pero creo que es implícita
69     if(!inscripcionDao.estaInscrito(participante, curso)){
70         //validamos que ese participante no se encuentre inscrito en 3 cursos
71         //esta es una regla de negocio explicita
72         if(inscripcionDao.getCantidadCursosPorParticipante(participante)<3){
73             //validamos que haya cupo
74             //esta es una regla de negocio explicita
75             if(inscripcionDao.hayVacantes(curso)){
76                 //creamos una instancia de inscripcion
77                 Inscripcion inscripcion = new Inscripcion(curso, participante, 0, ahora.getTime());
78                 //guardamos la inscripcion recuperando el id
79                 int nuevoId = inscripcionDao.guardar(inscripcion);
80                 //si todo ha sido exitoso mostramos el mensaje de inscripcion realizada
81                 JOptionPane.showMessageDialog(null, "Inscripcion realizada\n"
82                     + "Estimado "+participante.toString()+" ("+participante.getEmail()+") Usted se encuentra inscrito al curso"
83                     + "\nelegido con el número de inscripción "+nuevoId,
84                     "Informe de Inscripcion", JOptionPane.INFORMATION_MESSAGE);
85             }else{
86                 //creamos una instancia de inscripcion condicional
87                 Inscripcion inscripcion = new Inscripcion(curso, participante, 1, ahora.getTime());
88                 //guardamos la inscripcion en forma condicional e informamos la situación
89                 int nuevoId = inscripcionDao.guardar(inscripcion);
90                 //informamos de la situación
91                 JOptionPane.showMessageDialog(null, "Estimado "+participante.toString()+" ("+participante.getEmail()+") "
92                     + "En este momento no hay vacantes en el curso seleccionado.\n"
93                     + "Usted se encuentra inscrito en el curso elegido de forma condicional con el numero "+nuevoId
94                     + ".\nSi se producen vacantes, el Centro Cultural se pondrá en contacto con con usted para informarle."
95                     + "\n Muchas gracias.", "Informe de Inscripcion", JOptionPane.INFORMATION_MESSAGE);
96             }
97         }else{

```

Fin Caso de Uso