GESTIÓN Y MANEJO DEL SISTEMA DE APOYO ALIMENTARIO DE LA UNIVERSIDAD DISTRITAL

Daniel Garcia Perea 20141020212
Daniel Antonio Moreno Ramirez 20132020620
Edwar Diaz Ruiz 20141020004

8 de noviembre de 2017

Índice general

Parte I

Proyecto

Capítulo 1

Problema

Problema:
Dificultad que tienen los estudiantes para aprovechar el apoyo alimentario que ofrece la Universidad Distrital Francisco José de Caldas.

Objetivos:
Crear una aplicación que permita agilizar el proceso de entrega de almuerzo Mejorar el servicio de apoyo alimentario Automatizar el proceso registro de asistencia de usuarios

Pregunta: ¿Cómo agilizar el proceso de entrega de almuerzos en la Universidad ?

Hipotesis:
Una estrategia para agilizar el proceso de entrega de almuerzos en la universidad es la construcción de un aplicativo web que permita la entrega de papeles, organización de horarios y la verificación de asistencia al apoyo alimentario.

Es necesario que a través del aplicativo se pueda organizar horarios ya que si solo se gestiona la asistencia y entrega de papeles no habra una solucion al problema a largo plazo pero si gestionan los horarios estudiantiles se pueden crear estrategias para evitar el aglutinamiento de personas a la hora de recibir el almuerzo.

Justificación:
En la Universidad Distrital Francisco José de Caldas existe un serio problema de gestión en la distribución de almuerzo a los estudiante: la lentitud en la entrega ha ocasionado que los estudiantes no puedan aprovechar el servicio sin correr el riesgo de comprometer la llegada a sus clases. Muchos estudiantes necesitan el apoyo alimentario pero el llegar tarde a clase da lugar incidentes entre profesores y alumnos lo que puede afectar, a su vez, el desempeño académico

Capítulo 2

Metodología

2.1. Manifiesto Agil

Antes de decir que es Scrum, vale la pena saber que es el manifiesto agil y de que se compone.

El manifiesto agil se compone por 5 valores y 12 Principios:

2.1.1. Valores

Valorar a las personas y las interacciones entre ellas por sobre los procesos y las herramientas

Las personas son el principal factor de éxito de un proyecto de software. Es más importante construir un buen equipo que construir el contexto. Muchas veces se comete el error de construir primero el entorno de trabajo y esperar que el equipo se adapte automáticamente. Por el contrario, la agilidad propone crear el equipo y que éste construya su propio entorno y procesos en base a sus necesidades.

Valorar el software funcionando por sobre la documentación detallada

La regla a seguir es "no producir documentos a menos que sean necesarios de forma inmediata para tomar un decisión importante". Estos documentos

deben ser cortos y centrarse en lo esencial. La documentación (diseño, especificación técnica de un sistema) no es más que un resultado intermedio y su finalidad no es dar valor en forma directa al usuario o cliente del proyecto. Medir avance en función de resultados intermedios se convierte en una simple ïlusión de progreso".

Valorar la colaboración con el cliente por sobre la negociación de contratos

Se propone que exista una interacción constante entre el cliente y el equipo de desarrollo. Esta mutua colaboración será la que dicte la marcha del proyecto y asegure su éxito.

Valorar la respuesta a los cambios por sobre el seguimiento estricto de los planes

La habilidad de responder a los cambios que puedan surgir a lo largo del proyecto (cambios en los requisitos, en la tecnología, en el equipo, etc.) determina también su éxito o fracaso. Por lo tanto, la planificación no debe ser estricta sino flexible y abierta.

2.1.2. Principios

Los valores anteriores son los pilares sobre los cuales se construyen los doce principios del Manifiesto Ágil. De estos doce principios, los dos primeros son generales y resumen gran parte del espíritu ágil del desarrollo de software, mientras que los siguientes son más específicos y orientados al proceso o al equipo de desarrollo:

1. Nuestra mayor prioridad es satisfacer al cliente a través de entregas tempranas y frecuentes de software con valor.

2. Aceptar el cambio incluso en etapas tardías del desarrollo. Los procesos ágiles aprovechan los cambios para darle al cliente ventajas competitivas.

3. Entregar software funcionando en formafrecuente , desde un par de semanas a un par de meses, prefiriendo el periodo de tiempo más corto.

4. Expertos del negocio y desarrolladores deben trabajar juntos diariamente durante la ejecución del proyecto.

5. Construir proyectos en torno a personas motivadas, generándoles el ambiente necesario, atendiendo sus necesidades y confiando en que ellos van a poder hacer el trabajo.

6. La manera más eficiente y efectiva de compartir la información dentro de un equipo de desarrollo es la conversación cara a cara.

7. El software funcionando es la principal métrica de progreso.

8. Los procesos ágiles promueven el desarrollo sostenible. Los sponsors, desarrolladores y usuarios deben poder mantener un ritmo constante indefinidamente.

9. La atención continua a la excelencia técnica y buenos diseños incrementan la agilidad.

10. La simplicidad –el arte de maximizar la cantidad de trabajo no hecho– es esencial.

11. Las mejores arquitecturas, requerimientos y diseños emergen de equipos auto-organizados.

12. A intervalos regulares, el equipo reflexiona acerca de cómo convertirse en más efectivos, luego mejora y ajusta su comportamiento adecuadamente.

2.2. Scrum

Scrum es una de las metodologia agiles mas conocidas al igual que XP, es una herramienta en la cual se tienen varios grupos de trabajo enfocados en diversas tareas con un unico objetivo, en el cual se espera generar resultados y/o entregas al cliente en cortos periodos de tiempo.

El equipo de trabajo en esta metodologia esta apoyado por 2 roles: El ScrumMaster y el Product Owner. el ScrumMaster o tambien consederado Coach es aquel que vela por que se utilice Scrum , por remover las impedimentos y da asistencia al equipo para que logre el mayor nivel de rendimiento posible. El Product Owner es quien representa al negocio, stakeholders (trabajadores, organizaciones sociales, accionistas y proveedores, entre muchos otros actores clave), cliente y usuarios finales.

2.2.1. En que se basa

Esta basada en tres pilares:

Transparencia: Todos los implicados tienen conocimiento de qué ocurre y en el proyecto y cómo ocurre. Esto hace que haya un entendimiento "común" del proyecto, una visión global.

Inspección: Los miembros del equipo Scrum frecuentemente inspeccionan el progreso para detectar posibles problemas. La inspección no es un examen diario, sino una forma de saber que el trabajo fluye y que el equipo funciona de manera auto-organizada.

Adaptación: Cuando hay algo que cambiar, el equipo se ajusta para conseguir el objetivo del sprint. Esta es la clave para conseguir éxito en proyectos complejos, donde los requisitos son cambiantes o poco definidos y en donde la adaptación, la innovación, la complejidad y flexibilidad son fundamentales.

Parte II

UML

Capítulo 3

Análisis

3.1. Introducción

3.2. Diagrama de Casos de Uso

El diagrama de casos de uso es una forma de representar los requerimientos de un sistema, cada caso de uso reune una serie de requisitos basandose en diferemtes funciones o tareas.

Actor: Es una agrupacion uniforme de personas, sistemas o maquinas que interactuan con el sistema que estamos construyendo. Caso de Uso: Son secuencias de interacciones entre actores y un sistema que usan un servicio.

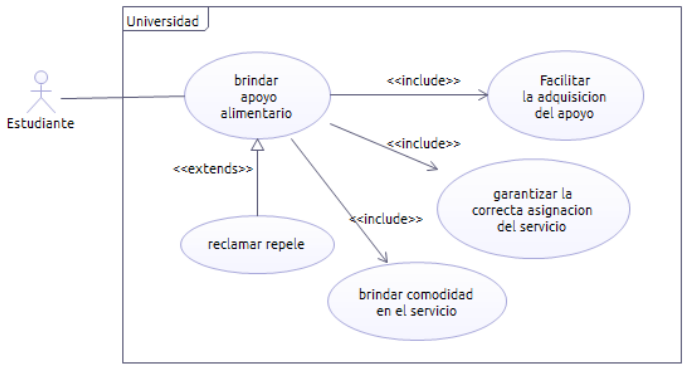

Figura 3.1: caso1

Cuadro 3.1: Caso de Uso CU-1

Nombre	Brindar Apoyo Alimentario
Actores	Estudiantes
Escenario	
Primario	Brindar Apoyo Alimentario
Secundario	No coinciden Horarios
Excepciones	Solicitud de Inscripción no aceptada, Estudiante no Registrado, Se acabo el alimento, Ser retirado del beneficio por alguna falla

Cuadro 3.2: Caso de Uso CU-2

Nombre	Reclamar Repele
Actores	Estudiantes
Escenario	
Primario	Reclamar Repele
Secundario	
Excepciones	No sobro alimento, Estudiante no Registrado, Ser retirado del beneficio, La hora del reclamo no es la adecuada

Cuadro 3.3: Caso de Uso CU-3

Nombre	Brindar Comodidad en el Servicio
Actores	Administrativos, Estudiantes
Escenario	
Primario	Brindar Comodidad en el Servicio
Secundario	Presentar Inscripción en la fechas no estipuladas
Excepciones	Se presenta algún error en la Infreestructura, No se abríran convocatorias No se cuentan con los espacios o horarios disponibles.

Cuadro 3.4: Casso de Uso CU-4

Nombre	Garantizar Correcta Asignación del Servicio
Actores	Empleados y Administrativos encargados del apoyo
Escenario	
Primario	Garantizar Correcta Asignación del Servicio
Secundario	Presentarse en las horas no adecuadas, Presentar alguna inconveniente en un requerimiento del Apoyo
Excepciones	Fallos en la Infreestructura, Falla del Personal encargado de brindar el servicio, Inconvenientes presentados tanto en espacios como horarios habilitados para la entrega del servicio

Cuadro 3.5: Caso de Uso CU-5

Nombre	Facilitar Adquisición del Apoyo
Actores	Administrativos, Personal encargado de brindar el servicio
Escenario	
Primario	Facilitar Adquisión del Apoyo
Secundario	Presentar alguna inconsistencia en la inscripción o documentos que soportan la misma,
Excepciones	Fallos en la Infreestructura, No poder optar por el beneficio por falta de cupos, No tener el puntaje necesarios para recibir el beneficio

3.2.1. Diagrama de Secuencia

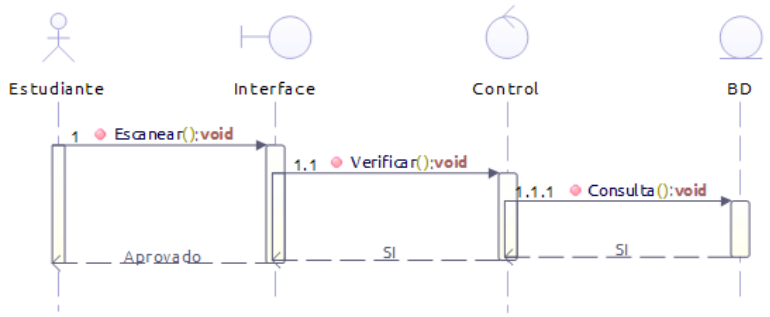

Figura 3.2: Diagrama de Asistencia

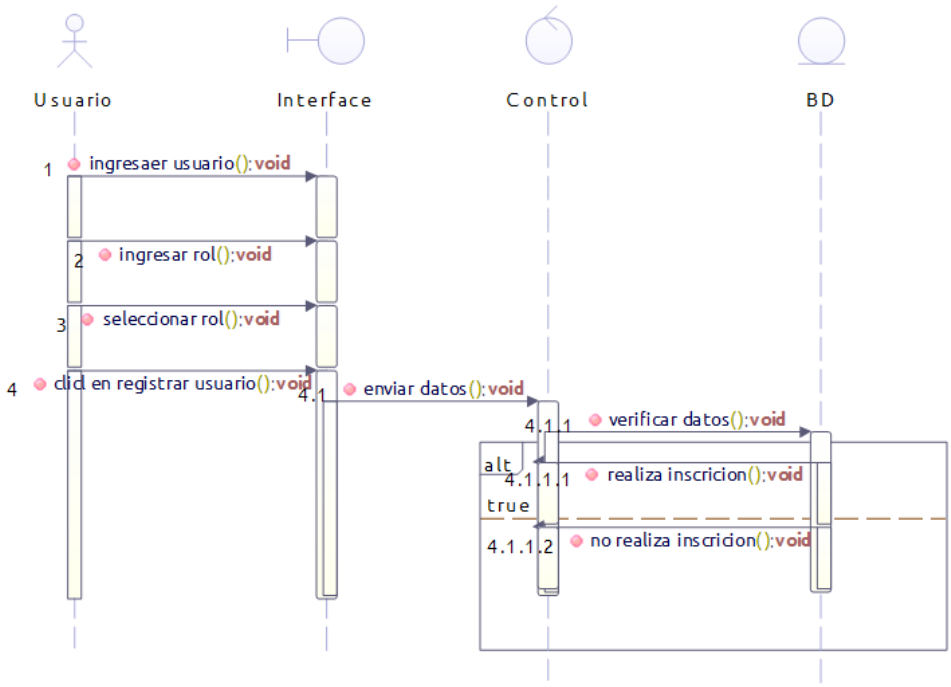

Figura 3.3: Registro del Usuario

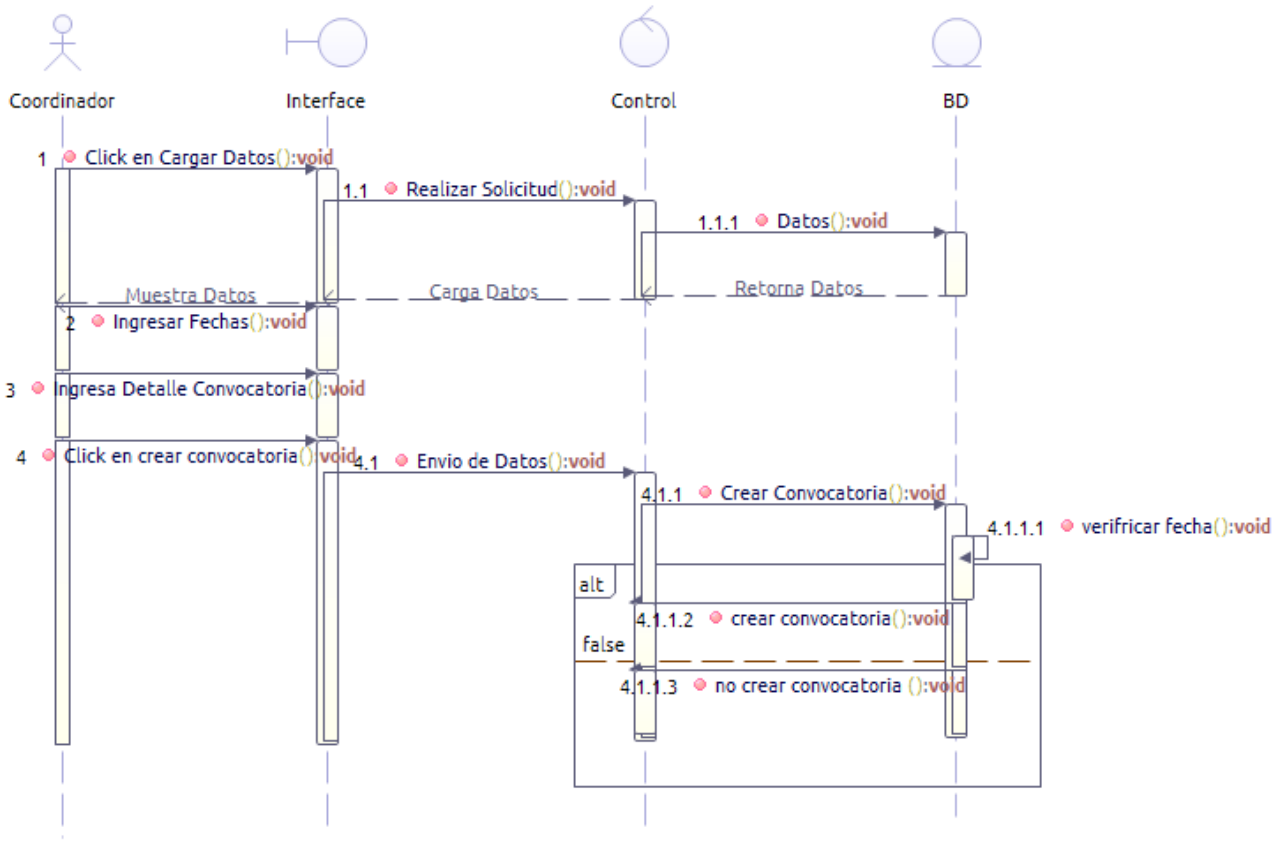

Figura 3.4: Crear convocatio del apoyo

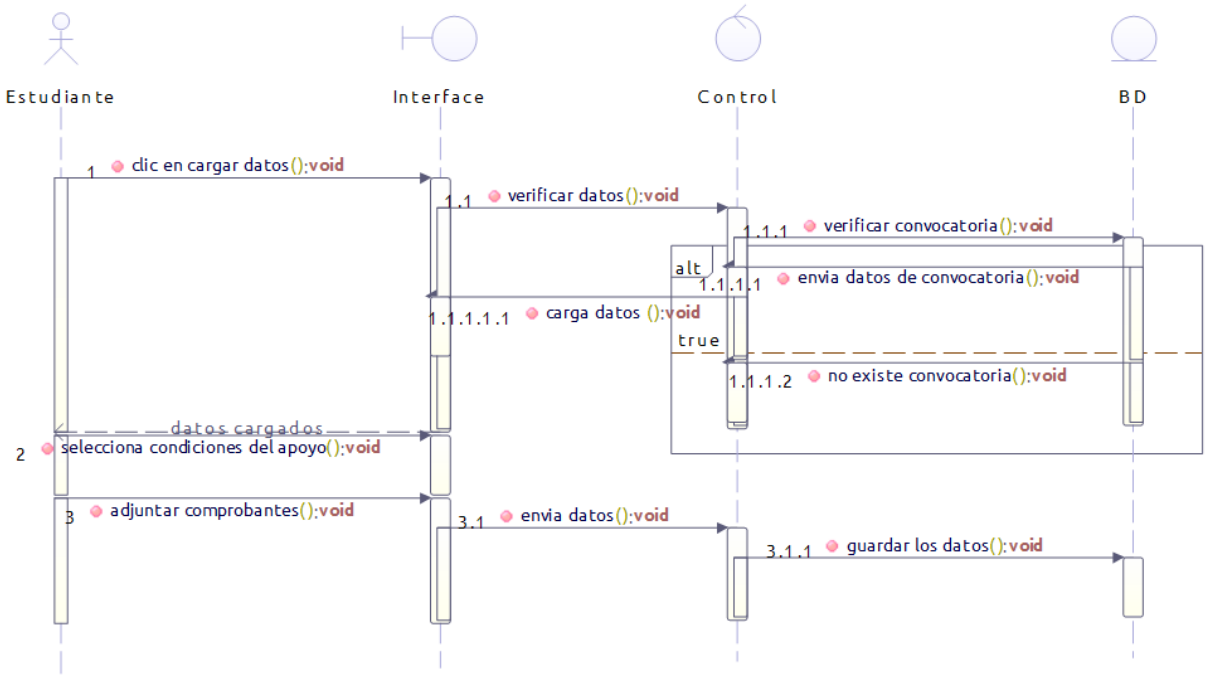

Figura 3.5: Solicitud para registrarse a una convocatoria

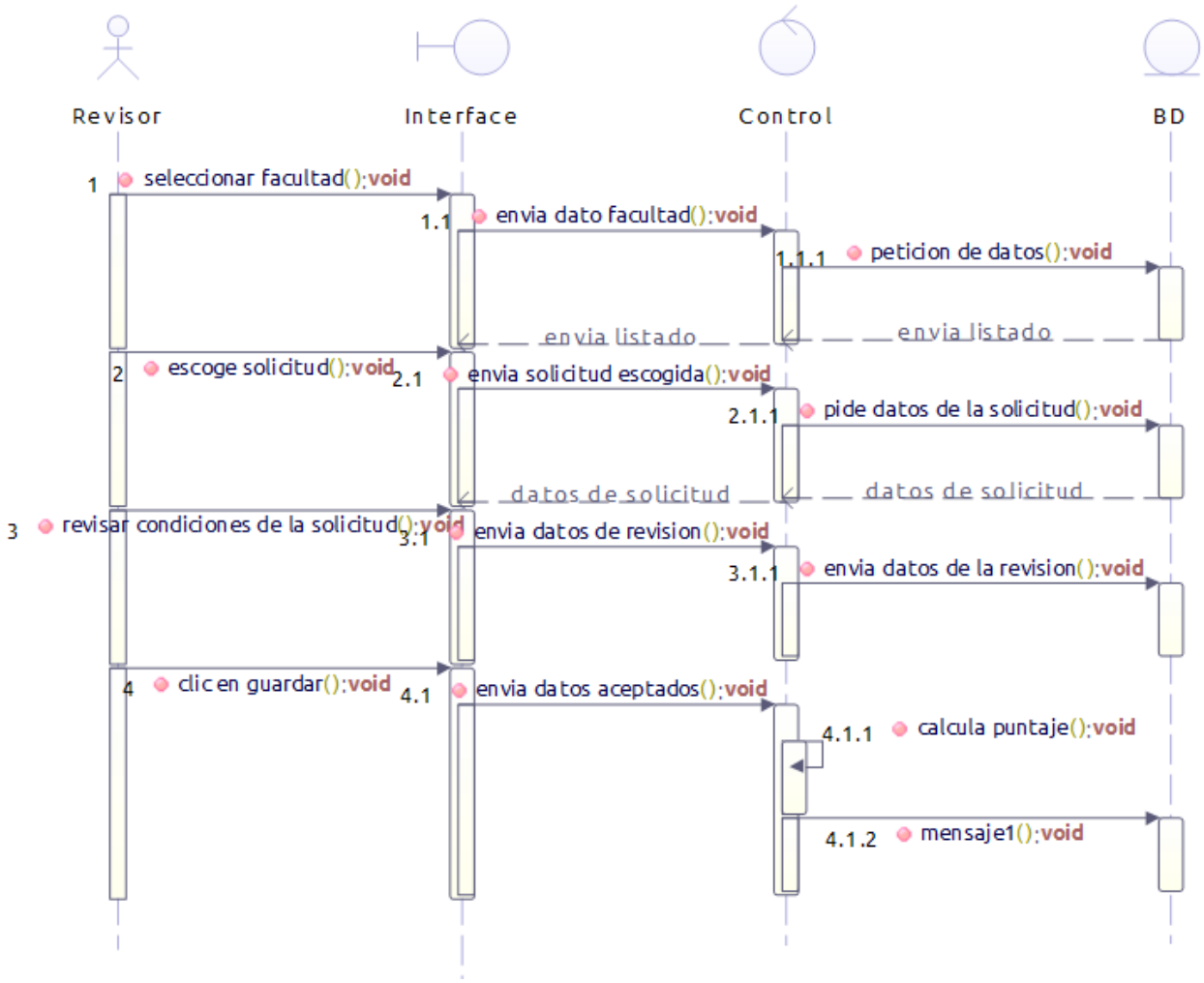

Figura 3.6: Verificar solicitud a una convocatoria

3.2.2. Diagrama de Comunicación

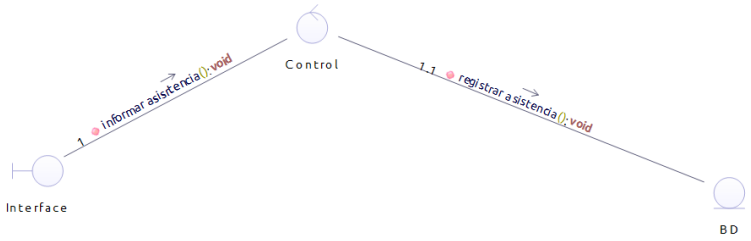

Figura 3.7: Diagrama de Asistencia

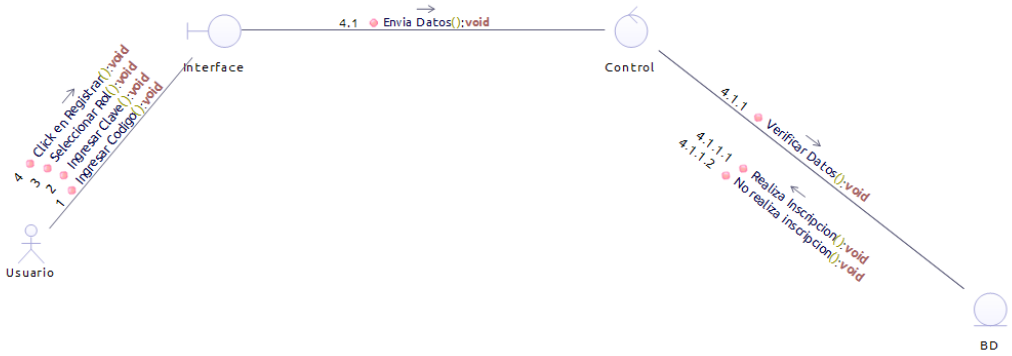

Figura 3.8: Registro del Usuario

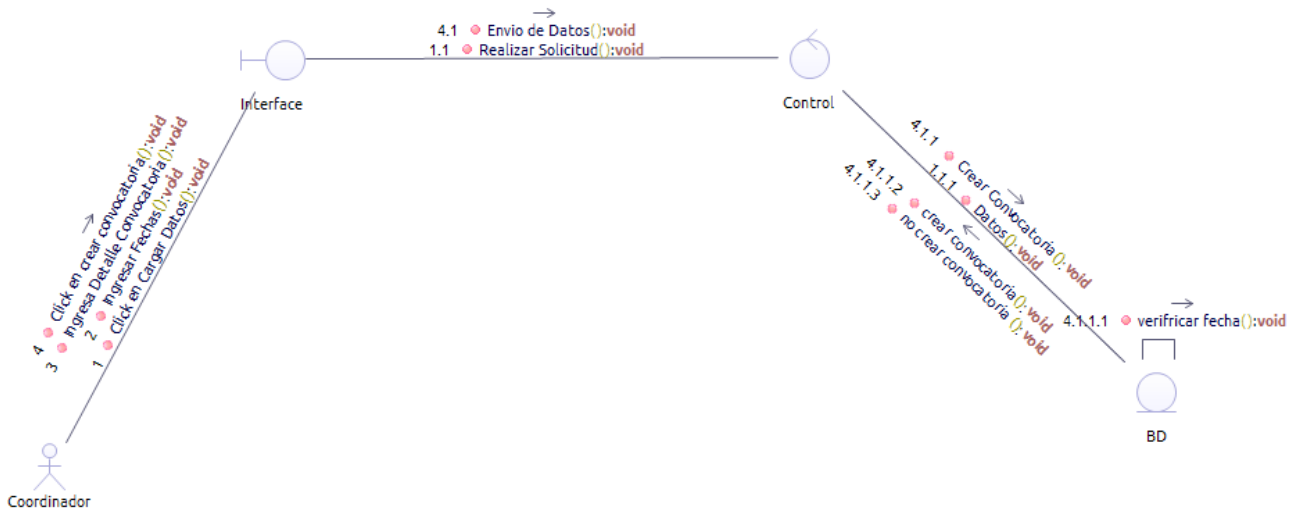

Figura 3.9: Crear convocatoria del apoyo

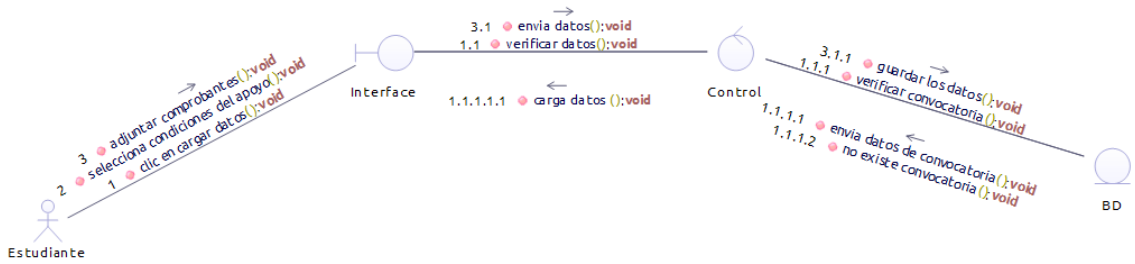

Figura 3.10: Solicitud para registrarse a una convocatoria

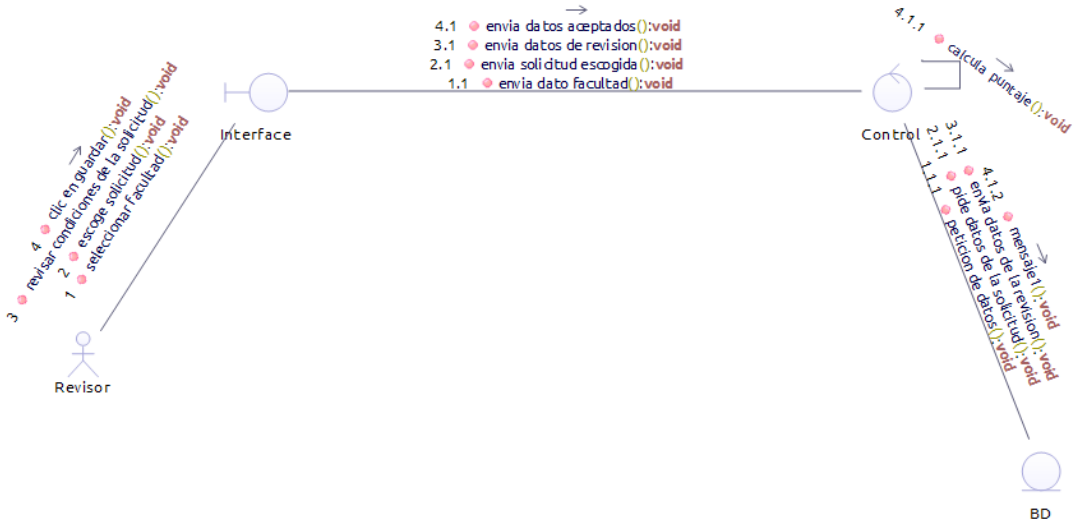

Figura 3.11: Verificar solicitud a una convocatoria

Capítulo 4

Diseño

4.1. Diagrama de Clases

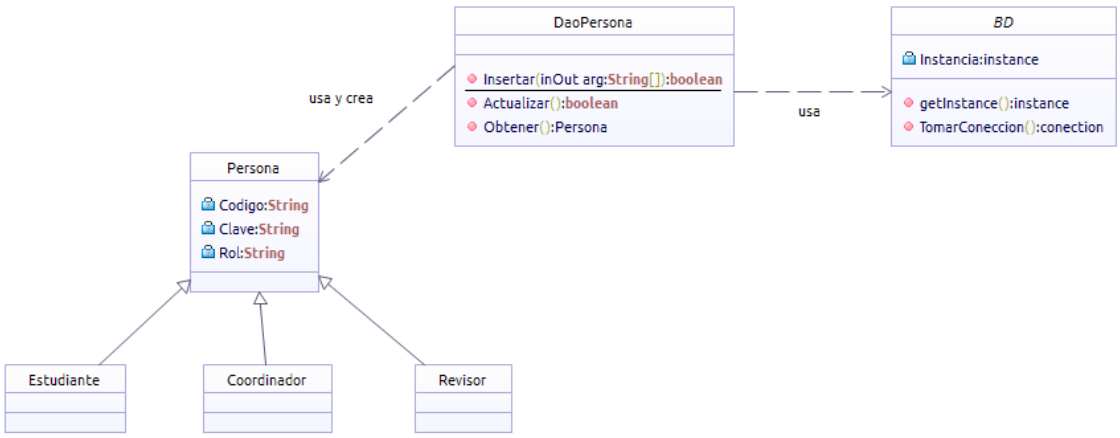

Figura 4.1: Registrar Usuario

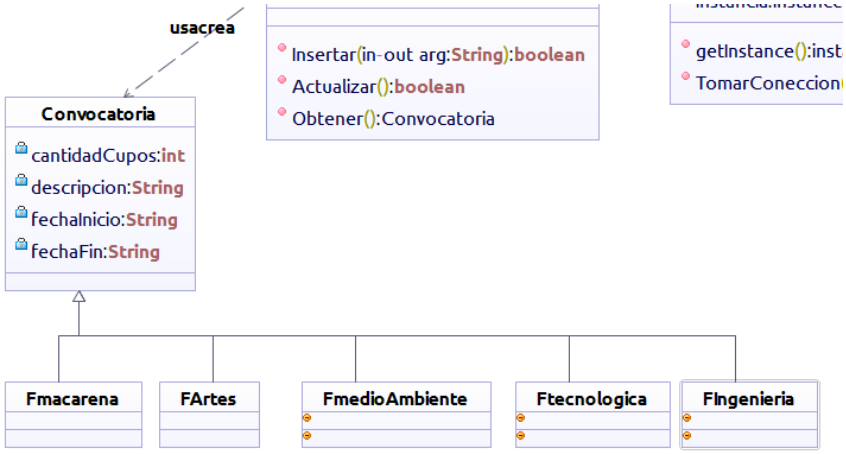

Figura 4.2: Registrar Convocatoria

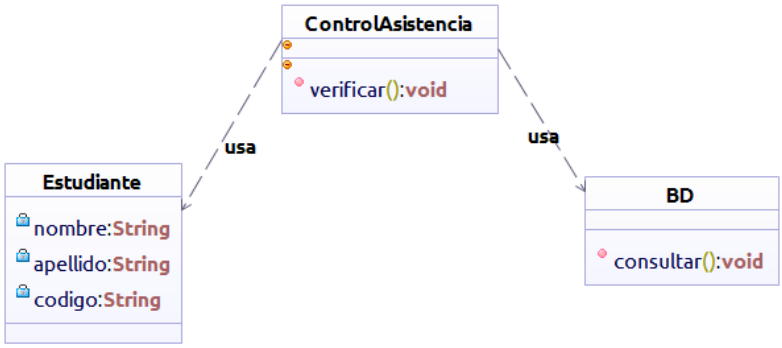

Figura 4.3: Asistencia

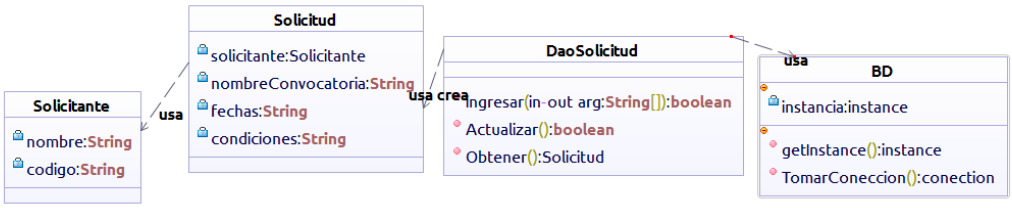

Figura 4.4: Registro Solicitudes

Capítulo 5

Patrones

5.1. Patrones

5.1.1. Creacionales

Builder

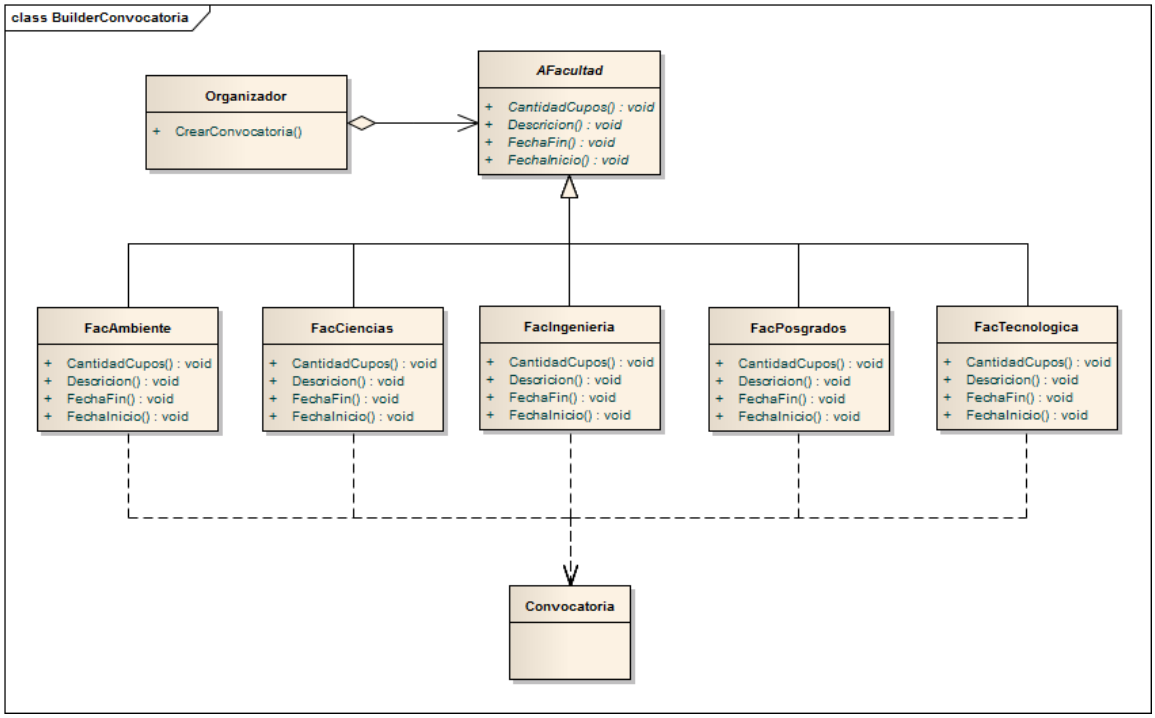

Figura 5.1: Builder para crear convocatoria

Factory

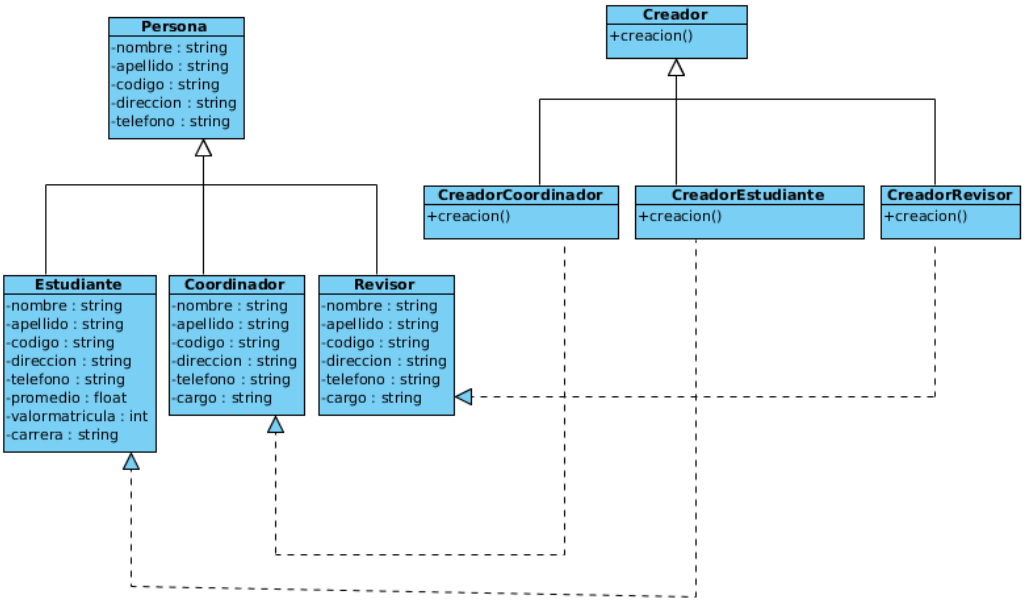

Figura 5.2: Fabrica para crear personas

Singleton

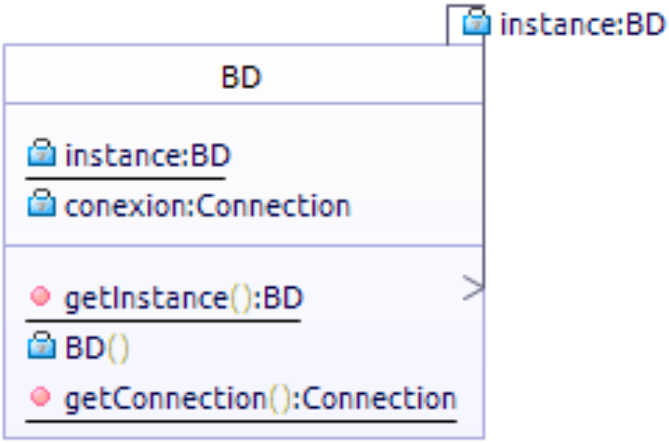

Figura 5.3: Singleton para conexion con la base de datos

5.1.2. Estructurales

5.1.3. Comportamiento

Iterador

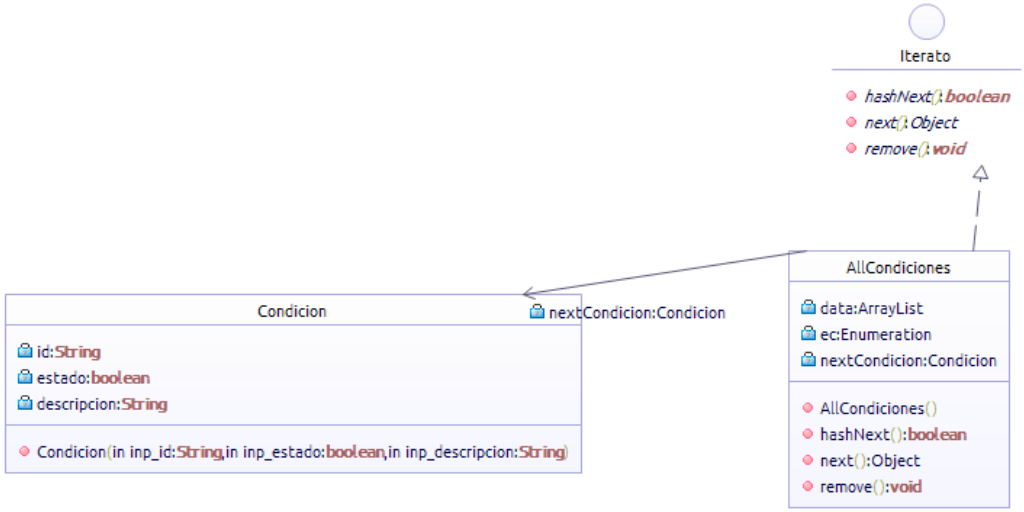

Figura 5.4: Iterador, muestra listado de condiciones para el apoyo

State

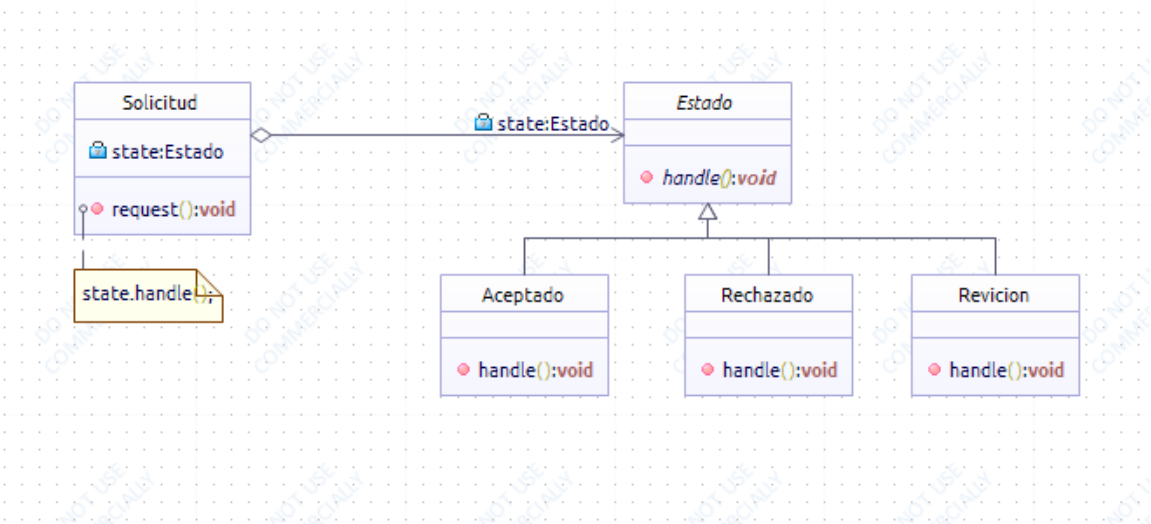

Figura 5.5: Estado, Muestra los estados de las solicitudes

Anexos

5.2. Codigo Patrones

5.2.1. Estado

```
          public class Solicitud{
        private Estado state;


        public void request(){
        state.handle();

}
}



public class Aceptado extends Estado{
public void handle(){}
}

public class Rechazado extends Estado{
public void handle(){}
}

public class Revicion extends Estado{
public void handle(){}
}
```

5.2.2. Builder

```
public abstract class AFacultad {
public AFacultad m_AFacultad;
public AFacultad(){
}
public void finalize() throws Throwable {
}
public abstract void CantidadCupos();
public abstract void Descripcion();
public abstract void FechaFin();
public abstract void FechaInicio();
}

public class Convocatoria {
public Convocatoria(){
}
public void finalize() throws Throwable {
}
}

public class FacAmbiente extends AFacultad {
```

```
public FacAmbiente(){

}

public void finalize() throws Throwable {
super.finalize();
}

public void CantidadCupos(){

}

public void Descripcion(){

}

public void FechaFin(){

}

public void FechaInicio(){

}

}



public class FacCiencias extends AFacultad {

public FacCiencias(){

}

public void finalize() throws Throwable {
super.finalize();
}

public void CantidadCupos(){
```

```
}

public void Descripcion(){

}

public void FechaFin(){

}

public void FechaInicio(){

}

}


public class FacIngenieria extends AFacultad {

public FacIngenieria(){

}

public void finalize() throws Throwable {
super.finalize();
}

public void CantidadCupos(){

}

public void Descripcion(){

}

public void FechaFin(){

}
```

```
public void FechaInicio(){

}

}


public class FacPosgrados extends AFacultad {

public FacPosgrados(){

}

public void finalize() throws Throwable {
super.finalize();
}

public void CantidadCupos(){

}

public void Descripcion(){

}

public void FechaFin(){

}

public void FechaInicio(){

}

}

public class FacTecnologica extends AFacultad {

public FacTecnologica(){

}
```

```
public void finalize() throws Throwable {
super.finalize();
}

public void CantidadCupos(){

}

public void Descripcion(){

}

public void FechaFin(){

}

public void FechaInicio(){

}

}


public class Organizador {

public AFacultad m_AFacultad;

public Organizador(){

}

public void finalize() throws Throwable {

}

}
```

5.2.3. Singleton

```
package ;
public class BD{
private static BD instance;
private Connection conexion;
public static BD getInstance(){}
private  BD(){}
public static Connection getConnection(){}
}
```

5.2.4. Iterator

```
package ;
public class AllCondiciones implements Iterato{
private ArrayList data;
private Enumeration ec;
private Condicion nextCondicion;
public  AllCondiciones(){}
public boolean hashNext(){}
public Object next(){}
public void remove(){}
}


package ;
public class Condicion{
private String id;
private boolean estado;
private String descripcion;
public  Condicion(String inp_id,boolean inp_estado,String inp_descripci
}


package ;
public interface Iterato{
/**

*/
boolean hashNext();
/**

*/
Object next();
/**

*/
void remove();
}
```

Bibliografía