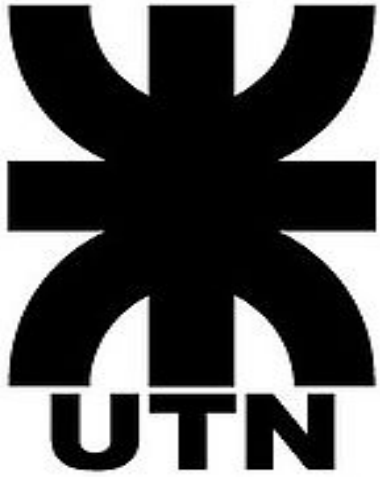


Universidad Tecnológica Nacional

Ingeniería en sistemas de información

Cátedra: Diseño de sistemas (DSI).

Docentes:

- Meles, Silvia Judith.
- Bene, Florencia.
- Andrade, Emiliano

Curso: 3k4

Grupo nro.: 9

Entrega nro.: "5. Implementacion".

Integrantes:

- Llamas, Franco Emmanuel. 69923. francollamas077@gmail.com
- Downes, Agustin. 70103. agustindownes@gmail.com
- Gallo, Gonzalo. 54784. gonzalo.hca@gmail.com
- Tosco, Rodrigo. 69404. rodrigotosco95@gmail.com
- Taborda, Matias. 54239
- Emanuel Márquez. 68294

Fecha Entrega: 08/11/19

Índice

Contenido

CONSIGNA

A continuación, se describen las presentaciones que se efectuarán a lo largo del año y las actividades a incluir en cada una:

1° Entrega:

Modelo de Dominio: construir el modelo de objetos del dominio, utilizando

un diagrama de clases, que contenga atributos y métodos para las clases; y para las relaciones la navegabilidad y multiplicidad.

2° Entrega:

- Realizaciones de casos de uso de análisis de 3 (tres) escenarios de casos de uso designados por el docente tutor. Se tomará como referencia únicamente las clases del Modelo de Dominio entregado por los docentes. Se modelará 1 escenario utilizando diagrama de comunicación y 2 escenarios utilizando diagrama de secuencia. Ver Anexo de Asignación de CU y Máquinas de Estado.
- Vista de Clases de Análisis utilizando un diagrama de clases. La vista debe incluir las clases de análisis necesarias para el modelado de los escenarios del punto anterios.
- Máquina de Estados de las clases asignadas a cada grupo, incluyendo en Vista de clases de análisis, la estructura que da soporte a la máquina de estados.

3° Entrega:

- Identificación y clasificación de Requerimientos No Funcionales.
- Patrones Arquitectónicos: Identificar tres patrones y modelar la aplicación de estos patrones a los problemas arquitectónicos que el grupo considere que requieren de su aplicación. Definir la aplicación y motivación de cada patrón a utilizar.
- Vistas de la Arquitectura.
- Vista de la funcionalidad: incluyendo justificación de la elección de los casos de uso que incluyen.
- Vista del diseño: Subsistemas e Interfaces o Vista de Despliegue: Nodos y componentes o Vista de Despliegue: Niveles de hardware.

4° Entrega:

- Rediseño de dos realizaciones de casos de uso de análisis, aplicando Patrones

de Diseño de Gamma de diferentes categorías. Para cada patrón defina:

  - Identificación: descripción del problema que resuelve.
  - Estructura: Vista de clases de diseño (especificación de tipos de datos,tipos de retornos y tipos de parámetros y visibilidad de métodos – privados, públicos, etc.) correspondiente a la estructura resultante de la aplicación del patrón.
  - Dinámica: Realización de casos de uso de diseño, utilizando un diagrama de secuencia para la parte dinámica; donde se modele la aplicación del patrón.
  - Pseudocódigo: código o descripción textual del comportamiento que se ejecuta en cada uno de los métodos que están afectados en la aplicación del patrón.

Los patrones se acuerdan con el docente tutor antes de la presentación.

- Diseño de Interfaz del caso de uso a implementar, indicando al menos un patrón de IHM descripto con la plantilla correspondiente.

5° Entrega:

- Implementación de un caso de uso en el que hayan diseñado aplicando un patrón de diseño.
- Mapeo a base de datos relacional: Modelo de Entidad Relación para las clases que se ven afectadas en la realización del CU elegido:
  - Todas las entidades deben incluir sus correspondientes claves primarias y foráneas, siempre que sean necesarias. o Las entidades que representan transacciones deberán estar completas, con todas sus propiedades (atributos).
- Requerimiento de cambio:
  - Vista de la estructura: modelo de dominio (con diagrama de clases), incluyendo las clases de dominio modificadas y las nuevas a partir de la aplicación del o los requerimientos de cambio. Todas las clases deben tener sus métodos y atributos de dominio.
  - Listado de casos de uso que se agregan con una breve descripción.
  - Listado de casos de uso que se modificación con una breve descripción del cambio.

ENUNCIADO

E.R.S.

ANÁLISIS

Realización de caso de uso de análisis. Caso de uso 17 “Registrar pedido”

Realización de caso de uso de análisis. Caso de uso 39 “Generar remito”

Realización de caso de uso de análisis. Caso de uso 85 “Generar informe de efectividad”

Máquina de estados. Clase “Remito”

ENTREGAS CON CORRECCIONES APLICADAS

MODELO DE DOMINIO

ARQUITECTURA

Listado y descripción de Requerimientos No Funcionales

Patrones arquitectónicos

N TIER

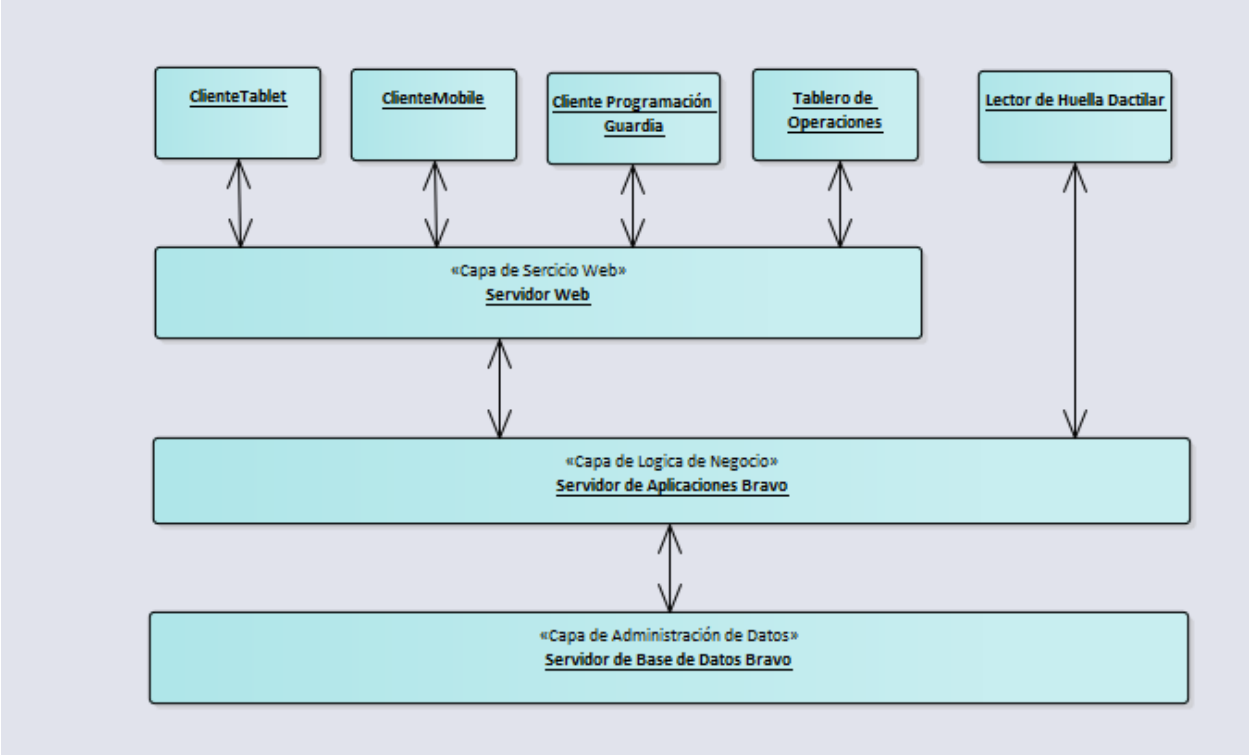
- Separación de intereses
- Comunicaciones sincrónicas
- Despliegue flexible

Motivación: La comunicación sincrónica entre capas brinda una mejor performance y confiabilidad en las transacciones.

La separación de intereses en distintas capas lógicas facilita la futura modificación y extensibilidad del sistema.

Nos permite utilizar clientes web delegados que no requieren demasiado hardware y también la distribución de la capa de datos en distintos servidores facilitando la redundancia ante eventuales catástrofes.

Aplicación: Corresponde a la vista de ejecución runtime que destaca sobre una estructura de capas, las comunicaciones entre las mismas y muestra la distribución de las capas en los niveles de hardware de la arquitectura.

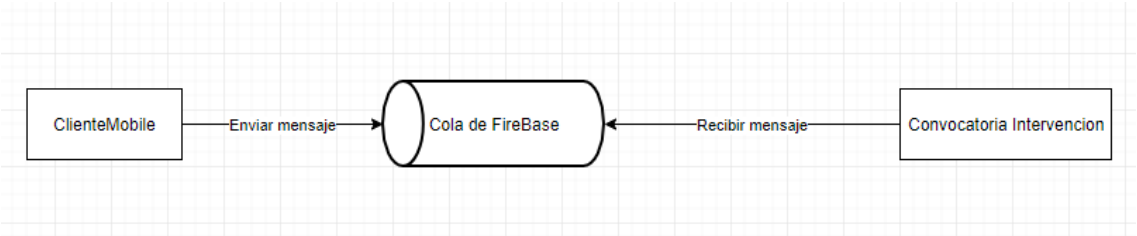


MESSAGING

- Comunicación asincrónica
- Calidad de servicio configurable
- Bajo acoplamiento

Motivación: A medida que el sistema solicita el envío de mensajes, se enviaran los mensajes lo más rápidamente posible en un caso critico como es la notificación a bomberos de llamado a intervenciones. Y se asegurara que llegue en caso de no estar disponible, el mensaje quedara en la cola hasta que sea recibido.

Aplicación: Para resolver él envió de mensajes vía whatsapp a un grupo de destinatarios aprovechando la potencialidad del patrón y las comunicaciones asincrónicas que soporta la falta eventual de comunicación. También se utiliza al implementar un servicio de Google firebase que maneja las colas de mensajes que se enviaran como notificaciones hasta que los mismos sean transmitidos

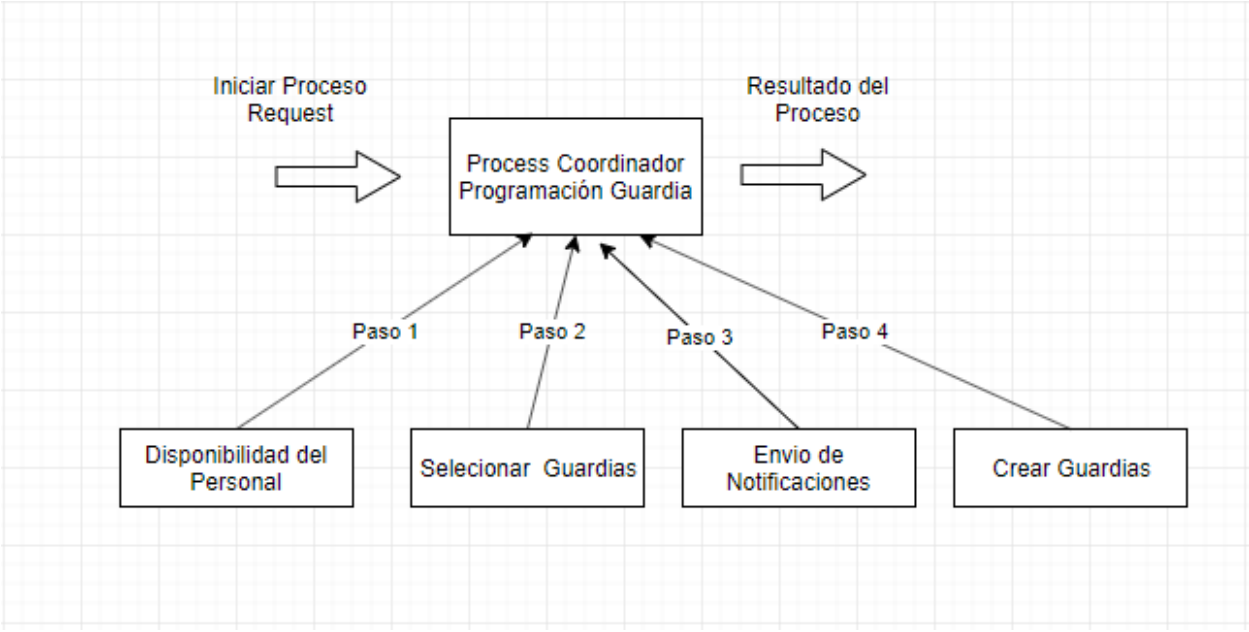


PROCESS COORDINATOR:

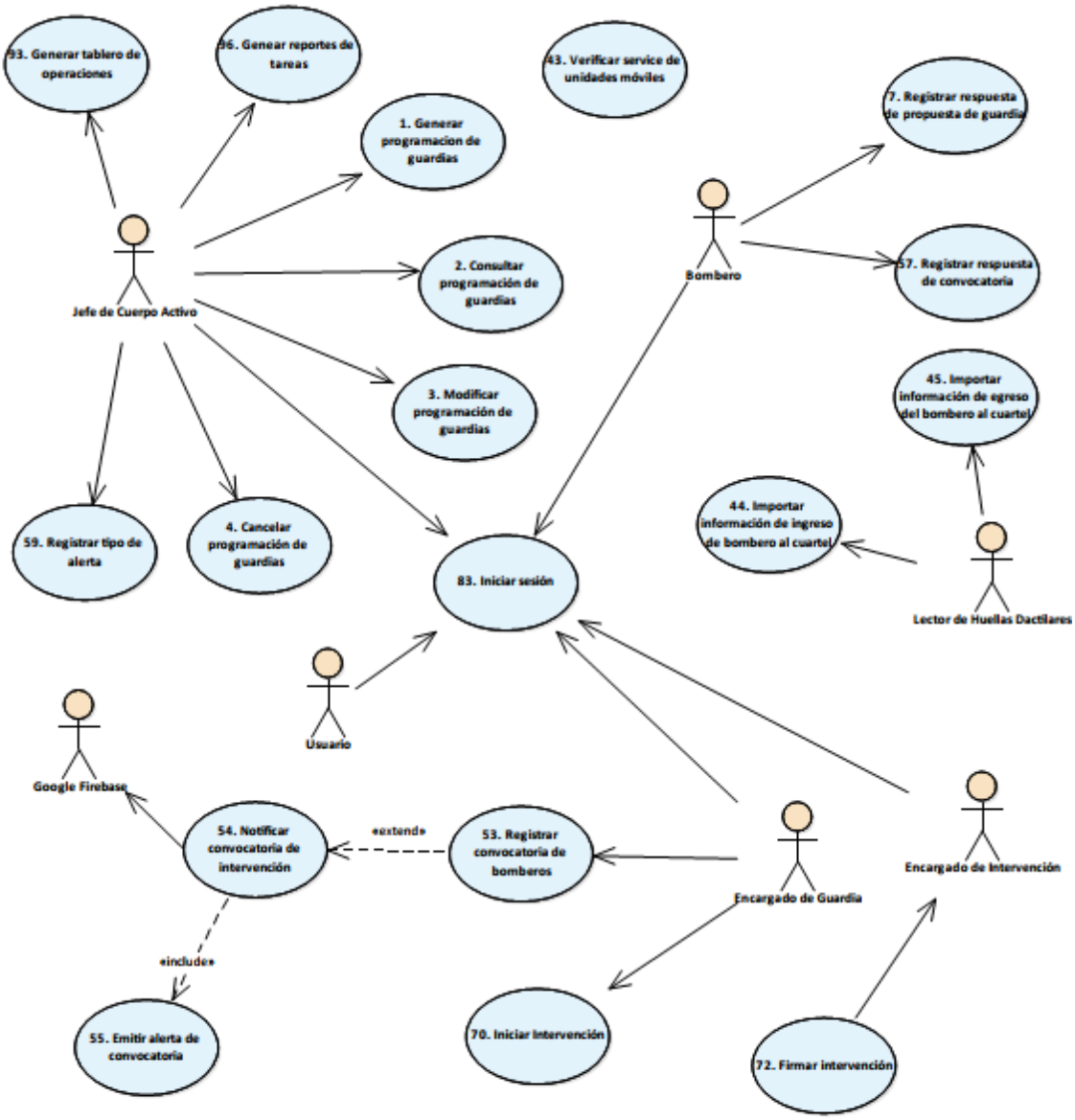
- Encapsulamiento del proceso
- Bajo acoplamiento
- Comunicación flexible

Motivación: Permite dividir un proceso en subprocesos que no se conocen entre sí sino que resuelven lo que les compete únicamente. Podremos así distribuir la carga en uno o más servidores y en caso de ser necesario modificar el código será mucho más fácil

Aplicación: Dividiremos el proceso de programación de guardias en subprocesos tales como: Disponibilidad del personal, edición de guardias, envío de notificaciones y creación de guardias con bomberos. Este patrón nos permite realizar este proceso independientemente de donde se encuentra distribuido cada componente.



Vista de la funcionalidad y justificación de casos de uso


93. Generar tablero de operaciones
96. Genear reportes de tareas
1. Generar programacion de guardias
43. Verificar service de unidades móviles
7. Registrar respuesta de propuesta de guardia
Jefe de Cuerpo Activo
2. Consultar programación de guardias
Bombero
57. Registrar respuesta de convocatoria
3. Modificar programación de guardias
45. Importar información de egreso del bombero al cuartel
59. Registrar tipo de alerta
4. Cancelar programación de guardias
83. Iniciar sesión
44. Importar información de ingreso de bombero al cuartel
Lector de Huellas Dactilares
Usuario
Google Firebase
54. Notificar convocatoria de intervención
«extend»
53. Registrar convocatoria de bomberos
Encargado de Guardia
Encargado de Intervención
«include»
55. Emitir alerta de convocatoria
70. Iniciar Intervención
72. Firmar intervención

Vista de la funcionalidad

Caso de Uso	Justificación
1. Generar programación de guardias, 2, 3, 4	Es la transacción más compleja y además cumple con todos los RNF relacionados al desarrollo web.
7. Registrar respuesta de propuesta de guardia	Relaciona al RNF 10, haciendo el uso de la Api de Google fireBase, que implementa el patrón messagin se soluciona el envió de notificaciones.
43. Verificar service de unidades móviles	El cual corre una tarea programada para responder al RNF 13.
44. Importar información de ingreso bombero al cuartel, 45	Es el encargado del procesamiento de la huella dactilar y se relaciona al RNF 12.
54. Notificar convocatoria de intervención	Relacionado al RNF 11, utilizando la API de Google FireBase
55. Emitir alerta de convocatoria	Relacionado al RNF 11, utilizando la API de Google FireBase
57. Registrar respuesta de convocatoria	Relacionado al RNF 9, que soluciona el envió y respuestas vía WhatsApp.
59. Registrar Tipo alerta, 60, 61, 62	Relacionado al RNF 19, soluciona el tipo de alerta. Notificación sonora convocatoria.
72. Firma intervención	Relacionado al RNF 21, firma digital.
83. Iniciar sesión, 84	Relacionado al RNF 15, interfaz de acceso personalizado.
93. Generar tablero de operaciones	Relacionado al RNF 17 y RNF 14, que hace que el tablero se refresque automáticamente.
96. Generar reportes de tareas, 94, 95, 97, 98	Relacionado al RNF 16, para la generación de reportes e informes.

Vista de diseño

Vista de despliegue (distribución de componentes)

Vista de despliegue (Hardware)

Patrones de Diseño

Builder

Identificacion

En el caso de uso generar informe de cumplimiento de guardias, pasado el paso 10, para la creación del reporte, esto nos permite separar la construcción del objeto de la presentación. En este caso se hace el objeto para la presentacion por la pantalla.

Estructura

Dinamica

Pseudocodigo

Para crear una **PantallaGeneracionInformeCumplimiento** se debe llama al método construir() del **DirectorInforme**.

Éste consta de la llamada a la creación del producto y luego le asigna el encabezado, cuerpo y pie.

Creación del producto:

Este método construir() llama al método polimorfico crearProducto() de **ConstructorInforme**, que en este caso **ConstructorConcretoPantalla** lo implementa de la siguiente manera:

```
metodo publico construirProducto(): void {
        informe = nuevo ProductoPantallaGeneracionInformeCumplimiento()
}
```

Este método varía en las otras implementaciones, creando el producto que le corresponde.

Luego para construir el Encabezado esta dinámica lo implementa:

```
metodo publico construirEncabezado(fechaDesde: Date, fechaHasta: Date): void {
        titulo = “Informe de cumplimiento de los bomberos de ” + fechaDesde + “ hasta “ +
fechaHasta

        informe.agregarTitulo()
}
```

Para la construcción del cuerpo y del pie se hace lo mismo, se procesa la info y se llama al método de la pantalla agregarCuerpo() y agregarPie() respectivamente

Para distintos tipos de Informe, esto se implementa de forma diferente.

Por último se llama al método obtenerProducto() de **ConstructorConcretoPantalla** que solo está definido en ese constructor concreto (debido a que el retorno es ese producto específico)

```
metodo publico obtenerProducto(): ProductoPantallaGeneracionInformeCumplimiento {
        devolver informe
}
```

STATE

Identificacion

Se aplica el CU Importar información de ingreso de bombero al cuartel para rediseñar el paso 5 y 6. Que el estado se encargue de resolver lo que indica el paso 6, donde se le delegan al estado la creacion de la asistencia.

Estructura

Dinamica

Pseudocodigo

Para verificar si el estado está en curso, se creó un método polimorfico esEnCurso() que en la implementación de la clase EnCurso va a devolver True, mientras que en el resto de los estados devolverá False.

Para la creación de una asistencia una vez que se encontró la guardia en curso para ese bombero, se llama al método crearAsistencia(fechaIngreso: Date) de la clase **GuardiaBombero.** Al bombero lo obtiene de su atributo.

Este método llama al método polimorfico crearAsistencia(bombero: Bombero, fechaIngreso: FechaIngreso) de **EstadoGuardia**.

La creación de la asistencia que anteriormente hubiera sido responsabilidad de la guardia es delegada al estado

```
metodo publico crearAsistencia(fechaIngreso: FechaIngreso): void {
        estado.crearAsistencia(this.bombero, fechaIngreso)
}
```

La implementación de este método en la clase **EnCurso** va a crear la asistencia y pasarsela por parámetro al método registrarIngreso() del bombero.

```
metodo publico crearAsistencia(bombero: Bombero, fechaIngreso: FechaIngreso): void {
        definir nuevaAsistencia: Asistencia
        nuevaAsistencia = nueva Asistencia(fechaIngreso)

        bombero.registrarIngreso(nuevaAsistencia)
}
```

En los demás estados no debería crear asistencia, por lo que en ellos se vería así:

```
metodo publico crearAsistencia(bombero: Bombero, fechaIngreso: FechaIngreso): void {
        lanzar excepcion "No se puede crear una asistencia para una guardia que no esta en curso"
}
```

Diseño de Interfaz Humano-Maquina.

Clasificación: List of Things	
Patrón (cuál):	Row Striping
Motivación (por qué):	Los bloques de colores suaves definen y delimitan la información contenida en ellos, incluso cuando no puede usar espacios en blanco para separar los datos en "fragmentos".
Aplicación (cómo):	Se elije un par de colores de baja saturación que tengan un valor similar pero no idéntico. (En otras palabras, uno debe ser un poco más oscuro que el otro).

Clasificación: Acciones y comando	
Patrón (cuál):	Prominent ‘Done’ button
Motivación (por qué):	Los botones más importantes deben verse distintos y ser fácilmente reconocibles por el usuario
Aplicación (cómo):	Se cuenta con un botón para regresar a la pagina anterior el cual, debe poder el usuario entenderlo. Cuando se informa al usuario, debe poder aceptar. También existe un botón para que pueda volver a imprimirlo si así lo desea. En todo momento el usuario puede cancelar o volver atrás.

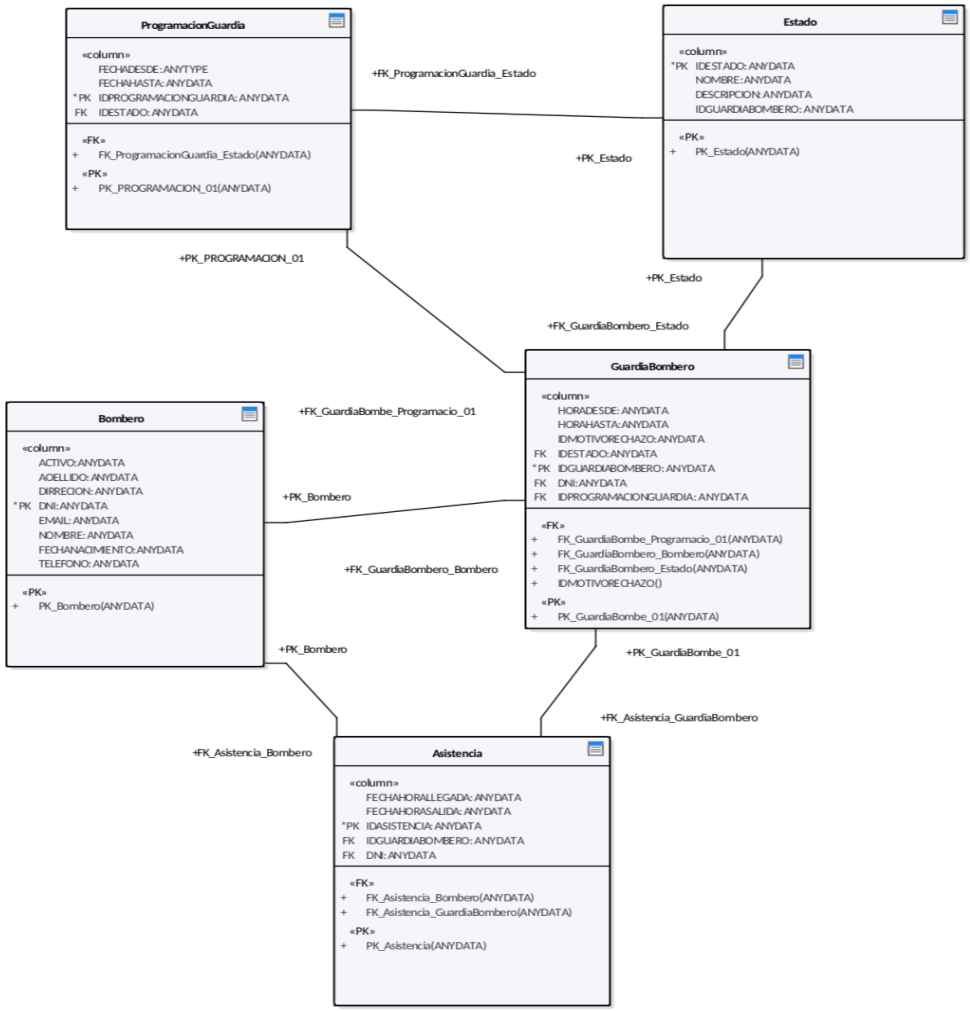
Clasificación: Acciones y comando	
Patrón (cuál):	DropDown Chooser
Motivación (por qué):	Permiten encapsulan interfaces de usuario complejas en un espacio pequeño. La página que ve el usuario sigue siendo simple y elegante, y la IU del selector solo se muestra cuando el usuario lo solicita, una forma adecuada de ocultar la complejidad hasta que sea necesaria.
Aplicación (cómo):	Se aplica cuando el usuario necesita seleccionar las

	fechas, de esta forma se despliega un menú calendario, pudiendo seleccionar el formato correcto, evitando que tenga que escribir el mismo.

Implementacion

Mapeo a Base de Datos Relacional.

Diagrama Entidad Relacion



Requerimiento de Cambio.

Casos de uso Agregados

1. Registrar Tipo de Guardia:
2. registra un nuevo tipo de guardia, asignando su nombre y descripción.
3. Modificar Tipo de Guardia: modifica los datos de un tipo de guardia existente.
4. Eliminar Tipo de Guardia**:** elimina un tipo de guardia.

5. Consultar tipo de guardia**:** obtiene la información de un tipo de guardia específico.

Casos de uso Modificados:

- Generar programación de guardias: al configurar las guardias de tipo “Bombero” para los bomberos, también se deberá configurar las guardias de tipo “Encargado” para los encargados. Para un bombero, en un día específico se podrán configurar varias guardias de tipo encargado, aunque solo una de tipo bombero (Ver Aclaracion 1)
- Importar información de ingreso del bombero al cuartel: se cambia el paso en el que corrobora que existan guardias en estado “EnCurso”, ya que esto dependerá de ahora en más del tipo de guardia. ○ Para guardias de tipo “Bombero” se mantiene el flujo habitual y chequea que exista una guardia de este tipo en curso, y la horaIngreso esté comprendida entre las ‘fechaHoraEntrada’ y ‘fechaHoraSalida’ ○ Para guardias de tipo “Encargado” chequea que exista una guardia de este tipo en estado “En Curso”, solo chequea que el día sea el mismo que el día de la fechaHoraEntrada o fechaHoraSalida. De esta forma, el encargado tiene más margen para llegar antes y ponerse operativo a tiempo.

Aclaracion 1: debido a que los turnos de encargado son de 24 horas y que el requerimiento de cambio indica que se mantenga el chequeo de disponibilidad, se toma que pueden haber guardias de tipo “Encargado” que duren 24 horas, como así también pequeñas guardias en diferentes momentos del día, cumpliendo así las 24 horas. De esta forma el puesto de un día podría turnarse entre varios encargados, pudiendo un bombero ser encargado más de una vez en un día.

Modelo de Dominio.