Ambito de decision (Objetos / Arquitectura / Persistencia / Otro)	Componente/s impactado/s	Decisión	Otras Alternativas	Justificación de la decisión
ENTREGA 1				
Arquitectura	Diagrama de Clases: Tramo Servicio Estacion Linea Servicio Publico de transporte	La comunidad tiene una lista de servicios que a su vez cada servicio tiene a que tramo pertenece. Cada tramo tiene a que estacion pertenece, cada estacion a que linea y cada linea a que servicio público.	- Los Servicios recidirian dentro de las Estaciones. - Que las comunidades conozcan servicios públicos por ende la linea de conocimientos seria inversa a la actual.	Nuestro sistema esta enfocado en conjuntos de comunidades con determinados servicios y problematicas de interés. De la otra forma, las comunidades estarían enfocadas en los medios de transporte y no en las problematicas.
Otro	Codigo	La clase "App" dentro del codigo es la que se ejecuta para hacer la prueba del validador. (Temporal)	- Utilizar un dispositivo de test.	En esta entrega consideramos que este tipo de testeo es suficiente. Esto se implementara en un futuro en el sistema. Esta decision es de caractere TEMPORAL.
Persistencia	Codigo / Diagrama de Clases	La clase "repo de usuario" para tener registro de los usuarios ingresados. No esta implementado en el codigo.	- Utilizar una base de datos	Las bases de datos que se utilizaran no son reflejadas en esta Entrega. Decidimos no utilizarlas en esta instancia.
ENTREGA 2				
Objetos	Diagrama de Clases: Establecimiento, Tramo	Abstraer la clase Tramo haciendo que los Establecimientos tengan una lista de Tramos	- Dejarlo unicamente en los servicios de transporte publico.	Anteriormente se habia diseñado tramo teniendo en cuenta únicamente los servicios de transporte públicos, pero no todas las entidades. Ese concepto de Tramo era insuficiente para el sistema
Arquitectura	Caso de Uso	Vinculamos el caso de uso de "Asociar Localizaciones" a Ciudadano y Entidad prestadora dado a que ambos interactuan en este.	- Separarlos en dos casos de uso distintos	En caso contrario se repetirian estructuras cuando es mejor que sean 2 por separados para una mejor implementacion a futuro.
Objetos	Diagrama de Clases: Establecimiento, Servicio	Que los Establecimientos tengan la lista lista de todos sus servicios.	-Que los servicios de un Establecimiento solo se puedan acceder a través de sus tramos.	Al acceder directamente a los servicios en lugar de pasar por cada tramo para obtenerlos, nos ahorramos acceso de memoria que podrían llegar a afectar al rendimiento si el sistema trabaja en gran escala.
Objetos	Diagrama de Clases	No se agregara para esta entrega los apartados relacionados con Organizacion.	-Agregarlo junto a toda informacion relacionada.	Preferimos no agregarlo a esta entrega ya que entro de las especificaciones falta informacion para el desarrollo.
ENTREGA 3				
Arquitectura	Caso de Uso	Vinculamos los casos de uso "Abrir un incidente", "Consultar estado del incidente" y "Cerrar un incidente" a Ciudadano dado que toda persona podrá interactuar con los incidentes	- Concentrar todas las acciones sobre los incidentes en un unico caso de uso	Consideramos que nos ayudaria a tener una mejor implementacion para el avance del proyecto

Objetos	Diagrama de Clases	Estrategia "compuesta" para el envío de notificaciones.	- Dentro de la estrategia de notificación según el tiempo, decidir el medio de envío con condicionales, o hacer una estrategia para cada modo y medio. Si lo hiciéramos de la segunda forma, nos permitiría definir reglas del estilo "por telegram no se mandan notificaciones inmediatas" (no habría una estrategia de NotificaciónAlToquePorTelegram).	El separar las estrategias de envío de notificaciones temporales y según el medio nos permite mayor extensibilidad en el caso de que aparezcan nuevas formas de notificación. Como nos parece difícil y con poco sentido que aparezcan reglas como las del contraejemplo, decidimos hacerlo así.
Objetos	Diagrama de Clases	Patrón composite para tratar los servicios como entidades "monitoreables"	- Hacer que los grupos de servicios sean listas de servicios.	Al usar este patrón de diseño, habilitamos la posibilidad de que existan "grupos de servicios compuestos", es decir, grupos de servicios dentro de grupos de servicios, y así poder clasificarlos más en profundidad según múltiples criterios. Además, nos permitiría agregar más entidades monitoreables con muchísima facilidad en el caso de que surja el requerimiento (surgió por el lado de las suscripciones a entidades).
Objetos	Diagrama de Clases	Método de notificación especial para incidentes oficiales	- Hacer un modulo de software que se encargue de hacer un polling para revisar constantemente si hay nuevas notificaciones oficiales para enviarlas.	Según el requerimiento especificado, todos los incidentes reportados oficialmente por una entidad, se notifican inmediatamente a los usuarios suscritos a esta. Por lo tanto, se ignoraría la estrategia de notificación de cada usuario. Hacer un método especial para esto nos permite hacer un "bypass" de la estrategia. De este modo también que se notifique al usuario como reacción directa a la carga del inicidente.
Objetos	Diagrama de Paquetes	Decidimos hacer el diagrama de paquetes y de clases juntos para una mejor comprensión de la distribucion de tareas y responsabilidades de cada sector	- Hacer ambos diagramas por separado, pero en el de Paquetes poner de forma mas especifica las agrupaciones logicas del mismo	Decidimos implementar ambos diagramas juntos para poder representar de forma mas completa y clara las distintas agrupaciones logicas que hay, con sus dependencias. Además nos permitió comprender mejor la consigna y tener una mejor compresión de como se conecta cada sector
Objetos	Diagrama de Componentes	Decidimos hacer que todos los Actores del diagrama de componentes dependan de la interfaz usuario	-No pensamos en otra manera para hacer una mejor dependencia con el sistema	ya que aun no esta implementada correctamente su interaccion con el sistema
Objetos	Diagrama de Componentes	La interfaz de usuario se encarga de la validacion, logueo, registro y todo lo que esta relacionado con la interfaz	-	Consideramos que es toda una logica enfocada al mismo componente