FUNDAMENTOS DE ARQUITECTURA DE SOFTWARE

Entregable #5

- Angel Santa Cruz Miñano
- Andre Puente Nuñez

Documento de Visión y Alcance

1. Introducción

El presente documento describe la visión y alcance del sistema. Este documento permite establecer el acuerdo inicial con el cliente acerca del desarrollo que se va a realizar. El contenido del documento es el siguiente:

1. Introducción

2. Contexto de negocio
   2.1 Antecedentes
   2.2 Frase del problema
   2.3 Objetivos del negocio

3. Visión de la solución
   3.1 Características del sistema

4. Alcance y limitaciones
   4.1 Alcance

5. Contexto del sistema
   5.1 Diseño de Arquitectura de Sistemas
   5.2 Diseño de Arquitectura de Software
   5.3 Diseño de Arquitectura de Aplicaciones
   5.4 Detección de Requisitos

6. QAW (Quality Attribute Workshop)
   6.1 Refinamiento de Escenarios

7. Introducción a DDD (Domain Driven Design)

8. Introducción a ADD (Attribute Driven Design)
   8.1 ¿Qué es ADD?
   8.2 Descripción del ciclo ADD
   8.3 Entradas y Salidas del método ADD
   8.4 Descomposición del sistema
   8.5 Interacción entre subsistemas
   8.6 Diagrama de Contexto
   8.7 Iteración #2
   8.8 Iteración #3
   8.9 Iteración #4

9. Diagrama caso de uso del Sistema

10. Diagrama de secuencia del Sistema

11. Resultados de ADD

12. Vista Modular

13. Vista Runtime

14. Vista Despliegue

15. Modelo de Datos

16. Sustento de eleccion web server y base de datos

2. Contexto de negocio

2.1 Antecedentes

RasBus es un sistema que ha sido pedido por la empresa MetroBUS.

2.2 Frase del problema

El sistema RasBus tiene como importante funcionalidad mostrar en una parada de buses, con visores, el tiempo estimado de llegada de los buses que paran en esa parada de buses.

2.3 Objetivos del negocio

Id	Descripción del objetivo del negocio
ON-1	Mostrar, en una parada de buses con visores, el tiempo estimado de llegada de los buses que paran en esa parada de buses.
ON-2	Permitir la integración del sistema con terceros que requieran información del bus más próximo a su parada.
ON-3	Mejorar de manera continua el servicio haciendo uso del sistema.

3. Visión de la solución

3.1 Características del Sistema

Id	Descripción	Prioridad	Objetivo del negocio asociado
C - 001	El sistema debe permitir el rastreo por GPS de todos sus buses para que puedan ser localizados en cualquier ubicación en un área de 100 metros.	**10**	**ON-1**
C - 002	El sistema debe proveer un tiempo estimado de llegada de los buses a cada una de las paradas principales.	**10**	**ON-1**
C - 003	El sistema debe recibir información del GPS, junto con el identificador del bus regularmente.	**10**	**ON-1**
C - 004	El visor debe mostrar cuatro o cinco buses y los tiempos asociados a estos buses.	**10**	**ON-1**
C - 005	El visor solo recibe información desde el sistema central.	**10**	**ON-1**
C - 006	*RasBus tiene un número de edificios donde un computador puede ser ubicado* y puede recibir la información del receptor de radio, vía Internet.	**10**	**ON-1**
C - 007	El sistema debe permitir almacenar datos históricos de los tiempos de viajes por 24 meses.	**5**	**ON-1, ON-3**
C - 008	El cliente puede revisar la información de RasBus en un aplicativo móvil que le indicará que paradero está más cerca según su ubicación y a qué hora llega el bus más próximo.	**5**	**ON-2**
C - 009	El sistema requiere las entradas al algoritmo de estimación, lo cual tiene implicaciones de almacenamiento (posición y velocidad); como también implicaciones de ancho de banda de comunicación.	**10**	**ON-1**
C - 010	El sistema debe poder estar sujeto a cambios como para la implementación de un sitio Web y/o accesos telefónicos.	**10**	**ON-3**
C - 011	El sistema debe tener un largo tiempo de vida (funcionamiento de larga durabilidad).	**10**	**ON-3**

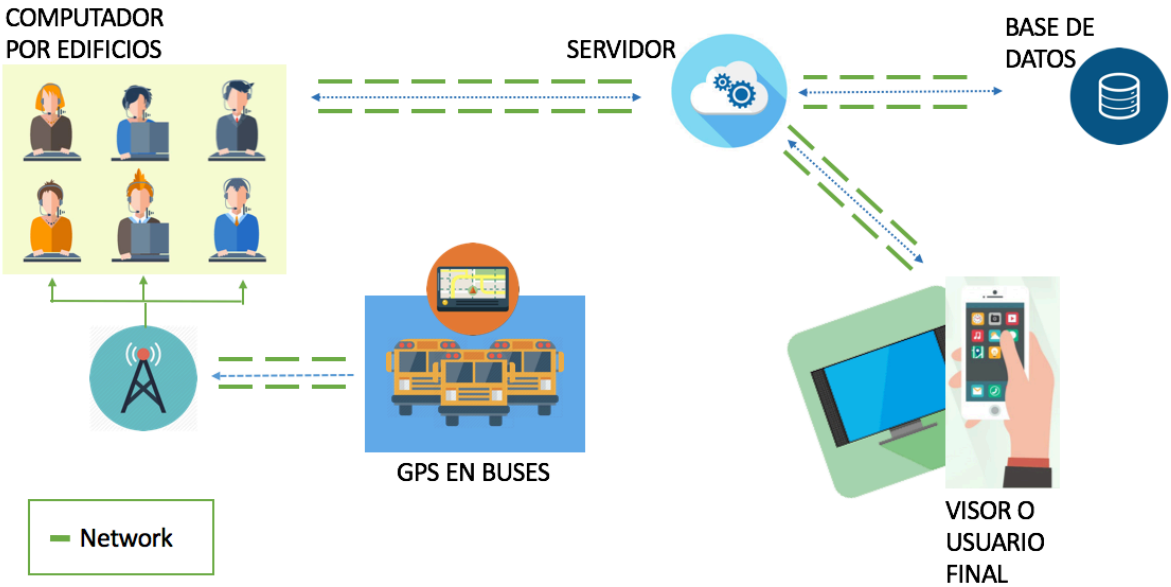
4. Alcance y limitaciones

4.1 Alcance

# de Release	Tema Principal
1.0	Funcionalidad Básica
2.0	Visibilidad
3.0	Estabilidad del sistema e integración con terceros

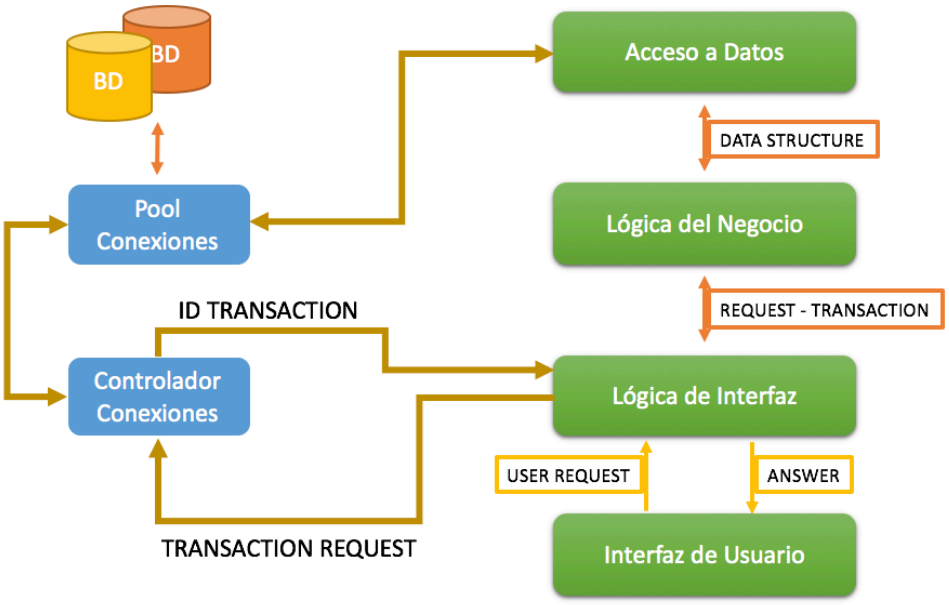
5. Contexto del Sistema

5.1 Diseño de Arquitectura de Sistemas

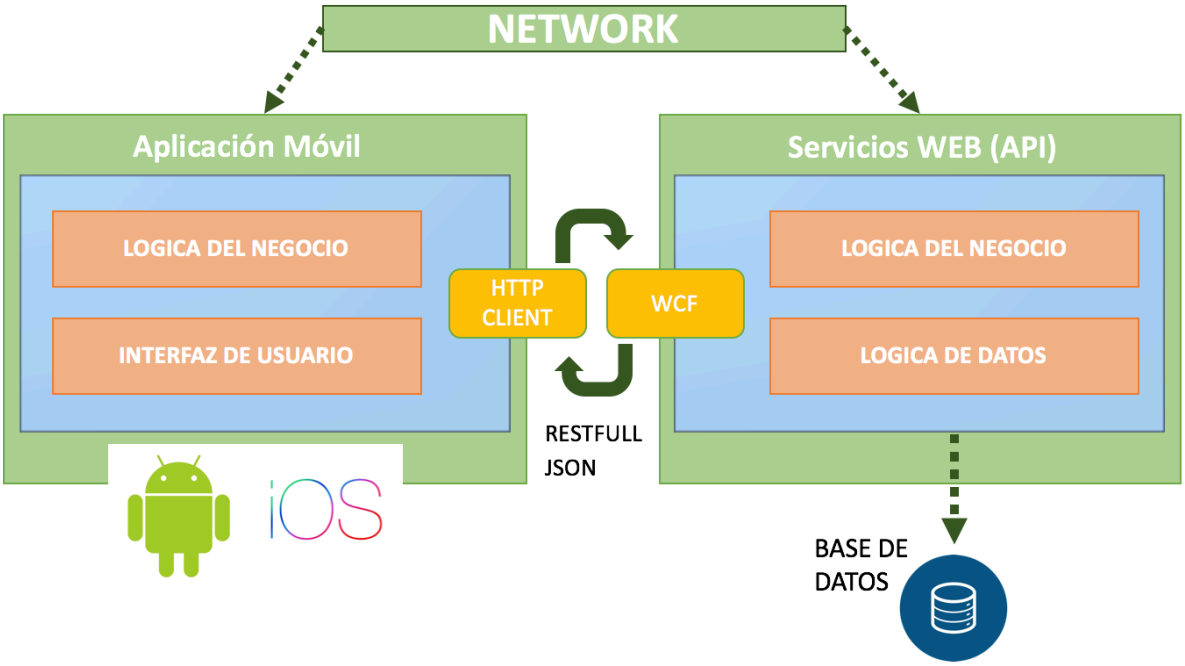


5.2 Diseño de Arquitectura de Software

SISTEMA:



APLICACIÓN MOVIL:



5.3 Diseño de Arquitectura de Aplicaciones

5.4 Detección de requisitos

- ✓ **Requisitos Funcionales:**
  - El sistema debe ser capaz de mostrar la información de llegada de un bus en la estación.
  - En el visor se debe mostrar la hora, el número de bus y el tiempo de llegada de los buses que lleguen a la estación en hora próxima.
  - El sistema debe de actualizar la posición y el tiempo de llegada del bus cuando éste haya tenido un percance (atraso, demora, retardo).
  - El sistema debe validar si el tiempo de llegada del bus está dentro o fuera del rango establecido y mostrar en pantalla su situación.
  - El sistema debe ser capaz de estimar el tiempo que se demora un bus en un paradero y añadirlo al visor.
  - El sistema debe listar a todos los buses que paren en un paradero.
  - El sistema debe dar el tiempo de llegada para el siguiente bus después del tiempo de solicitud en una ruta dada en una parada específica.
  - El sistema debe permitir al cliente poder revisar la información de RasBus en un aplicativo móvil que le indicará que paradero está más cerca según su ubicación y a qué hora llega el bus más próximo.

- ✓ **Requisitos No Funcionales:**
  - El Rendimiento: Tipo de procesador, algoritmo de estimación (posición y velocidad), ancho de banda de comunicación, tamaño de almacenamiento.
  - La Capacidad de Modificabilidad en línea telefónica, los sitios web y aplicaciones móviles.
  - Escalabilidad, El sistema puede crecer en el tiempo.
  - Disponibilidad, si la información que un visor recibe toma más de un minuto para desplazarse, entonces se va a pedir menos de una vez por minuto. Si se tarda menos de un minuto, entonces se volverá a mostrar en vez de actualizarla. (Se asume que no habrá tanta información para mostrar para que se cause un problema. Se supone que hay 40 informaciones de buses para mostrar. El visor puede mostrar 5 de ellos al tiempo. Si este desplaza uno hacia arriba cada segundo, se tomará 45 segundos para mostrarlos todos al menos 5 segundos (1 segundo cada fila).
  - Restricción, capacidad de 1000 buses en 100 paradas.

6. QAW (Quality Attribute Workshop)

6.1 Refinamiento de Escenarios

Refinamiento de Escenarios		
Escenario:		El sistema debe permitir localizar desde cualquier ubicación los buses más próximos a los 100 metros a la redonda.
Metas del Negocio:		ON-1
Atributos de calidad Relevantes:		Disponibilidad
Componente de escenarios	Estimulo:	Localizar desde cualquier punto los buses más próximos a los 100 metros a la redonda.
	Fuente de Estimulo:	Cliente
	Ambiente:	Sistema en normal funcionamiento
	Artefacto:	Sistema
	Respuesta:	Buses próximos a los 100 metros a la redonda localizados correctamente .
	Medida de Respuesta:	Listado de los 5 buses más próximos y tiempo de llegada estimado es actualizado en los visores y pantallas de usuarios.
Preguntas:		- ¿Qué condiciones podrían afectar a la disponibilidad? - ¿Cómo reacciona el GPRS ante un uso constante de captura y envió de datos? - ¿Cómo reacciona el Sistema ante un uso constante de captura y envió de datos?
Otras consideraciones:		En un ambiente de colección de datos como este será necesario implementar procedimientos basado en las buenas prácticas para el buen desempeño del sistema.
CONCERNS **CLASS OF SERVICE:** En este caso la disponibilidad está relacionada a cada visor que ciertamente tienen la misma funcionalidad pero cada uno opera independientemente del otro.		***TACTICAS*** **FAULT PREVENTION/** Se pueden prevenir errores **monitoreando los procesos.**

<table>
<tr><td colspan="3">Refinamiento de escenarios</td></tr>
<tr><td colspan="2">Escenario:</td><td>El computador de cada edificio del MetroBus puede recibir la información del receptor de radio solo vía Internet.</td></tr>
<tr><td colspan="2">Metas del Negocio:</td><td>ON-1</td></tr>
<tr><td colspan="2">Atributos de calidad Relevantes:</td><td>Disponibilidad</td></tr>
<tr><td rowspan="5">Componente de escenarios</td><td>Estimulo:</td><td>Recibir la información del receptor de radio vía Internet.</td></tr>
<tr><td>Fuente de Estimulo:</td><td>Cliente</td></tr>
<tr><td>Ambiente:</td><td>Receptor de radio y el computador de cada edificio tienen un normal funcionamiento.</td></tr>
<tr><td>Artefacto:</td><td>Receptor de radio, computador</td></tr>
<tr><td>Respuesta:</td><td>El computador de cada edificio recibe de manera exitosa la información del receptor de radio.</td></tr>
<tr><td></td><td>Medida de Respuesta:</td><td>El computador de cada edificio recibe la información del receptor de radio solo vía Internet.</td></tr>
<tr><td colspan="2">Preguntas:</td><td>- ¿Qué pasa si se requiere recibir la información por otra vía?
- ¿Cuál es el funcionamiento desde un punto a otro punto para lograr recibir la información?</td></tr>
<tr><td colspan="2">Otras consideraciones:</td><td>Estrategias de Interoperabilidad adecuada para soportar integración con terceros.</td></tr>
<tr><td colspan="2"><u>CONCERNS</u>

DOWNTIME:
Mantenimientos planeados en horarios bajos en atención.</td><td><u>TACTICAS</u>

Recovery-Preparation and Repair/
En casos como estos que dependemos de un hardware integrado a nuestro sistema, debemos contar con repuestos del mismo que permitan solucionar problemas de manera inmediata.</td></tr>
</table>

Refinamiento de escenarios		
Escenario:		Si el bus está a 1 kilómetro del visor, el tiempo estimado de llegada debe estar entre 2 minutos del tiempo actual, con una probabilidad del 95%
Metas del Negocio:		ON-1
Atributos de calidad Relevantes:		Rendimiento y Disponibilidad
Componente de escenarios	Estimulo:	Calculo de tiempo estimado de llegada de bus a 1 kilómetro del visor.
	Fuente de Estimulo:	Cliente
	Ambiente:	Sistema funcionando normalmente
	Artefacto:	No se conoce.
	Respuesta:	Si el bus está a 1 kilómetro del visor, el tiempo estimado de llegada debe estar entre 2 minutos del tiempo actual.
	Medida de Respuesta:	El cálculo de tiempo estimado de llegada del bus opera al 95%
Preguntas:		- ¿Qué pasa si el tiempo estimado de llegada esta entre más de 2 minutos del tiempo actual?
Otras consideraciones:		Estrategia de **Modificabilidad** adecuadas para poder mejorar el **Rendimiento** del sistema.
CONCERNS **RESPONSE TIME:** El tiempo estimado de llegada debe estar entre 2 minutos del tiempo actual.		***TACTICAS*** **Resource Demand/** Para lograr saber que 1km equivale a 2 minutos del tiempo actual se debe desarrollar un algoritmo que me permita **Incrementar la eficiencia informáticamente** para de esta manera mantener una alta probabilidad de que este opere entre el 95% y 100%.

Refinamiento de escenarios		
Escenario:		El sistema debe permitir mantener datos históricos de los tiempos de viajes por solo 24 meses.
Metas del Negocio:		ON-1
Atributos de calidad Relevantes:		Disponibilidad
Componente de escenarios	Estimulo:	Almacenar información de tiempo de viajes.
	Fuente de Estimulo:	RasBus - empresa
	Ambiente:	Sistema funcionando normalmente.
	Artefacto:	El sistema
	Respuesta:	Almacenamiento de datos históricos de los tiempos de viajes.
	Medida de Respuesta:	Los datos históricos de los tiempos de viajes serán almacenados solo por 24 meses.
Preguntas:		- ¿Qué pasa si los datos históricos de los tiempos de viajes son almacenados por más de 24 meses? - ¿Qué pasa si los datos históricos de los tiempos de viajes son almacenados por menos de 24 meses?
Otras consideraciones:		Usar estrategias de **Escalabilidad** adecuadas en caso de almacenar por más tiempo los datos históricos de los tiempos de viajes.
CONCERNS **DISASTER RECOVERY:** Es necesario ser recuperable ante cualquier desastre o mal funcionamiento de la BD.		***TACTICAS*** **Recovery Reintroduction/** Debemos contar con otra BD que cumpla con la función de hacer BACK-UP de la BD principal cada cierto tiempo para así mantener disponible la información en caso de cualquier falla**. (Passive redundancy)**

Refinamiento de Escenarios		
Escenario:		El sistema requiere las entradas al algoritmo de estimación, lo cual tiene implicaciones de almacenamiento (posición y velocidad); como también implicaciones de ancho de banda de comunicación.
Metas del Negocio:		ON-1
Atributos de calidad Relevantes:		Rendimiento, Disponibilidad
Componente de escenarios	Estimulo:	Implementación de las entradas al algoritmo de estimación.
	Fuente de Estimulo:	El cliente
	Ambiente:	Sistema funcionando normalmente
	Artefacto:	El sistema
	Respuesta:	Algoritmos de estimación operan de manera exitosa.
	Medida de Respuesta:	Los algoritmos de estimación implican almacenamiento y ancho de banda de comunicación.
Preguntas:		- ¿Qué puede provocar la degradación de la normalidad de trabajo del algoritmo? - ¿Qué artefactos del sistema pueden afectar el desempeño del algoritmo? - ¿El algoritmo es capaz de ser saturado al hacer uso constante del mismo? - ¿Qué artefactos del sistema se ven comprometidos debido al mal trabajo del algoritmo?
Otras consideraciones		Cada funcionalidad incorrecta del algoritmo debe ser grabada en un LOG y notificar a los operadores correspondientes la falla del mismo. Por otro lado, este error no puedo continuar dándose hasta un máximo de tiempo.
CONCERNS **Latencias y throughputs:** El aspecto técnico que más afecta a la percepción de usuario es la Latencia, recordemos que ellos desean una respuesta lo más pronto posible.		***TACTICAS*** **Resource Arbitration/** Podemos manejar el ancho de banda y troughputs por medio de **Políticas de planificación.**

Refinamiento de Escenarios		
Escenario:		El sistema debe poder estar sujeto a cambios como para la implementación de un sitio Web y/o accesos telefónicos.
Metas del Negocio:		ON-1 ON-2 ON-3
Atributos de calidad Relevantes:		Modificabilidad, Interoperabilidad
Componente de escenarios	Estimulo:	El sistema debe estar sujeto a cambios
	Fuente de Estimulo:	El cliente
	Ambiente:	El sistema función normalmente
	Artefacto:	El sistema
	Respuesta:	Nuevos interfaces de servicios disponibles.
	Medida de Respuesta:	Implementación de un sitio Web y/o accesos telefónicos.
Preguntas:		- ¿Qué tanto esfuerzo demanda la implementación del sitio web haciendo uso del sistema? - ¿Qué artefactos se ven afectados ante el despliegue del sitio web haciendo uso del sistema?
Otras consideraciones		Usar estrategias de **Interoperabilidad** adecuadas para poder integrar los nuevos interfaces de servicios con el sistema.
CONCERNS **Magnitud y dimensión del cambio y costo del cambio**		***TACTICAS*** Por otro lado, también, podemos **restringir caminos de comunicación** que nos permite integrar módulos, de esta manera mantenemos la coherencia semántica. **Mantener coherencia semántica/** Los cambios no deben afectar y deben tener un costo menor a la modificación de otros módulos.

Refinamiento de Escenarios		
Escenario:		El sistema debe tener un largo tiempo de vida (funcionamiento de larga durabilidad).
Metas del Negocio:		ON-3
Atributos de calidad Relevantes:		Rendimiento, Disponibilidad
Componente de escenarios	Estimulo:	Tiempo de vida del sistema
	Fuente de Estimulo:	El cliente
	Ambiente:	El sistema funciona normalmente
	Artefacto:	El sistema
	Respuesta:	El sistema debe tener un funcionamiento de larga durabilidad
	Medida de Respuesta:	Uso constante de **Deadlines** para un trabajo más organizado y cumplir a tiempo las entregas pactadas respecto a la mejora del proyecto.
Preguntas:		- ¿Qué artefactos del sistema se ven afectados debido a la continua operatividad a largo plazo? - ¿Qué porcentaje en costos representa el tiempo de vida de estos? - ¿El mal desempeño de estos artefactos produce un mal desempeño en otros artefactos?
Otras consideraciones		Siempre es necesario considerar el tiempo de vida de los artefactos a usar para la construcción o desarrollo de un sistema y evaluar como el mal desempeño de estos involucra el tiempo de vida de otros artefactos. Por ello es necesario establecer fechas de mantenimiento y de esta manera garantizar el mejor desempeño del sistema.
CONCERNS **Response y thinking time:** El sistema debe mantener un tiempo de respuesta optimo a las operaciones que se realizan en ella.		***TACTICAS*** **Resource demand – management/** Se deben **manejar tasas de eventos** e **incrementar los recursos disponibles** para mantener optimo el sistema.

<table>
<tr><td colspan="3">Refinamiento de Escenarios</td></tr>
<tr><td colspan="2">Escenario:</td><td>El visor puede mostrar 5 informaciones de los buses al mismo tiempo. Si este desplaza uno hacia arriba cada segundo, se tomará 45 segundos para mostrarlos todos al menos 5 segundos</td></tr>
<tr><td colspan="2">Metas del Negocio:</td><td>ON 1</td></tr>
<tr><td colspan="2">Atributos de calidad Relevantes:</td><td>Rendimiento, Disponibilidad</td></tr>
<tr><td rowspan="5">Componente de escenarios</td><td>Estimulo:</td><td>El visor lista los 5 buses más próximos y la información correspondiente a cada uno.</td></tr>
<tr><td>Fuente de Estimulo:</td><td>El cliente</td></tr>
<tr><td>Ambiente:</td><td>El sistema funciona normalmente</td></tr>
<tr><td>Artefacto:</td><td>El visor</td></tr>
<tr><td>Respuesta:</td><td>El visor muestra 5 informaciones de buses al mismo tiempo</td></tr>
<tr><td>Medida de Respuesta:</td><td>Si este desplaza uno hacia arriba cada segundo, se tomará 45 segundos para mostrarlos todos al menos 5 segundos</td></tr>
<tr><td colspan="2">Preguntas:</td><td>- ¿Qué pasa si el visor tarda más de 45 segundos en mostrar todos?
- ¿Cómo afecta a la empresa el retraso de la información?</td></tr>
<tr><td colspan="2">Otras consideraciones</td><td>Al detectar errores en la muestra de información esta se guarda en un LOG y alerta el mal funcionamiento. Además, es importante que se le de mantenimiento al sistema para que no exista retardo a la hora de enviar las informaciones.</td></tr>
<tr><td colspan="2"><u>CONCERNS</u>

Tasa de fallas:
Se puede decir que la fiabilidad de un sistema es la probabilidad de que ese sistema funcione o desarrolle una cierta función, bajo condiciones fijadas y durante un período determinado.</td><td><u>TACTICAS</u>

Faul Detection/
Podemos hacer uso de HeartBeat donde un componente emite un mensaje de latido periódicamente y si este falla se notifica que hace falta la corrección de fallos.</td></tr>
</table>

7. Introducción a DDD (Domain Driven Design)

7.1 ¿Qué es Domain Drive Design?[1]

Domain Drive Design (Diseño guiado por el dominio en español) se denomina como un enfoque para el desarrollo de software para las necesidades complejas mediante la conexión profundamente la implementación de un modelo de evolución de los conceptos centrales del negocio.
El DDD no es una tecnología ni una metodología, este provee una estructura de prácticas y terminologías para tomar decisiones de diseño que enfoquen y aceleren el manejo de dominios complejos en los proyectos de software.
En muchos proyectos la mayor complejidad es debida al dominio del problema, y es ésta complejidad la que trata de atajar DDD, partiendo de las premisas de que el foco principal del proyecto está sobre el dominio y que el diseño de un dominio complejo se basa en un modelo. DDD está orientado a las metodologías ágiles y supone que el desarrollo es iterativo y que existe una estrecha relación entre los desarrolladores y los expertos del dominio.

7.2 ¿Qué es lenguaje Ubicuo?

El lenguaje ubicuo es un término que introdujo Eric Evans en su libro sobre DDD (Domain Driven Design) como propuesta para crear un lenguaje común entre los programadores y los usuarios.
La definición propone nombrar las variables, métodos y clases con lenguaje del dominio de modo que sea "autoexplicable".
Con este tipo de nombres, el código se documenta por sí mismo:
"getRandomEntityFrom($modelName)" devuelve una entidad aleatoria a partir del nombre de un modelo.
Comprueba si el modelo existe en la colección antes de acceder a él.
"createMultipleEntities($numberOfEntities)" una cantidad de entidades determinada por el parámetro que recibe.
Para ello usa un bucle en el que se crea cada vez una nueva entidad a partir del contador actual.
Habitualmente los nombres representan objetos y los verbos métodos.

[1] http://tratandodeentenderlo.blogspot.pe/2013/08/domain-driven-design.html

7.3 ¿Qué son Capas de la Arquitectura(Layers Architecture)?[2]

Se proponen cuatro capas conceptuales:

- ***Interface de usuario (User Interface)***

  Responsable de presentar la información al usuario, interpretar sus acciones y enviarlas a la aplicación.

- ***Aplicación (Application)***

  Responsable de coordinar todos los elementos de la aplicación. No contiene lógica de negocio ni mantiene el estado de los objetos de negocio. Es responsable de mantener el estado de la aplicación y del flujo de esta.

- ***Dominio (Domain)***

  Contiene la información sobre el Dominio. Es el núcleo de la parte de la aplicación que contiene las reglas de negocio. Es responsable de mantener el estado de los objetos de negocio. (La persistencia de estos objetos se delega en la capa de infraestructura.

- ***Infraestructura (Infraestructure)***

  Capa de soporte para el resto de capas.

  - ✓ Provee la comunicación entre las otras capas.
  - ✓ Implementa la persistencia de los objetos de negocio
  - ✓ Implementa las librerías de soporte para las otras capas (Interface, Comunicación, Almacenamiento, etc.…).

Debido a que son capas conceptuales, su implementación es variada y en una misma aplicación se tendrá partes o componentes que formen parte de estas capas.

7.4 ¿Qué son Asociaciones (Associations)?

Un modelo típicamente tiene muchas asociaciones que puede hacer que la aplicación y el mantenimiento complicado (especialmente muchos a muchas asociaciones).
Haciendo asociaciones más manejables
- Imponer una dirección de recorrido
- Añadir un partido de clasificación
- Eliminar las asociaciones no esenciales

Esto hace que las asociaciones sean más expresivas de las del modelo, así como más manejables.

[2] http://dddcommunity.org/

7.5 ¿Qué son Entidades (Entities)?[3]

Cualquier objeto del dominio que mantiene un estado y comportamiento más allá de la ejecución de la aplicación y que necesita ser distinguido de otro que tenga las mismas propiedades y comportamientos, es una Entidad. Por ende, se le debe asignar un identificador único. Entonces, se puede definir identidad a todos los objetos de la aplicación que tengan un identificador único.
Qué se considere entidad dependerá principalmente de la función, objetivo y contexto de la aplicación. Habitualmente las entidades serán objetos con identidad propia en el mundo real.
Ejemplo:
En el mundo real, dos personas con el mismo nombre, edad y características, son dos personas distintas y por ello se inventó el Documento de Identidad, o DNI, Pasaporte, etc.
Para mí, los árboles no representan entidades, dado que no podría reconocer uno de otro. Pero para un jardinero encargado de catalogar todos los árboles de un parque, si la tienen y por ello su primera función será crear un método de clave única para identificar cada árbol.
Por tanto, Entidad es un término conceptual cuya concreción a la hora de definir en nuestra aplicación qué objetos o clases se tratarán como tales, dependerá del contexto y los objetivos de esta.

7.6 ¿Qué son Objetos de valor (Value Objects)?

Cuando un objeto no llega a ser entidad, se le considera un objeto de valor. Los objetos de valor dentro de una aplicación son indistinguibles unos de otros. Consisten únicamente en un conjunto de propiedades y comportamientos, pero no mantienen identidad alguna. Sí es posible que mantengan un estado, pero este se pierde al terminar la aplicación.
Características:

1. Se pueden duplicar y destruir con facilidad. Y no deberían ser modificados una vez han sido creados.
2. Pueden proporcionarse a capas fuera del dominio, dado que su posible modificación no afectaría a la aplicación.
3. Pueden contener a su vez otros objetos de valor.
4. En ocasiones es posible agrupar varios atributos en un objeto de valor.

Ejemplos:

- Dirección de un usuario podría ser un objeto de valor que contenga: Calle, Ciudad, Estado, País. A su vez, la calle podría ser un objeto de valor que contenga: Nombre de la calle, número del portal, piso, escalera, letra, etc.
- Un número de teléfono podría componerse de prefijo internacional y número. (Y compañía telefónica)

[3] https://github.com/jatubio/5minutos_laravel/wiki/Resumen-sobre-DDD.-Domain-Driven-Design

7.7 ¿Qué son Servicios (Services)?

La mayoría de los verbos del lenguaje ubicuo se convertirán en métodos de los objetos de la capa de negocios. Pero hay verbos que no son fáciles de concretar a cuál de los objetos corresponden. Estos verbos o comportamientos se llaman Servicios.
Ejemplo:

- La reparación del automóvil, ¿A quién corresponde?, ¿Al usuario/conductor o al vehículo?
- Cuando envío un paquete postal, ¿Quién hace el envío? ¿El remitente o el destinatario?

En realidad, son "Servicios" que contratamos, porque no se implementa, sino se usa.
Sin embargo, si soy el dueño de un taller de vehículos o una empresa de mensajería, esos servicios son el núcleo de mi aplicación y de mi modelo de negocio.
Los servicios actúan como interface o medio de relacionarse de varios objetos. Se puede tener servicios en cualquiera de las capas de la aplicación.
¿Cómo distinguirlos?

- Son comportamientos que no pertenecen a ninguna entidad.
- Pueden intercambiarse con facilidad por servicios de similares características.
- Van a ser usados por varias entidades en distintos puntos de la aplicación.
- Nos interesa más el resultado del proceso que el proceso en sí.
- Desde nuestro punto de vista, no tienen un estado interno (o nos es desconocido y no nos interesa).

7.8 ¿Qué son Módulos(Modules)?

Los módulos dan a la gente dos vistas del modelo: Pueden mirar detalle dentro de un módulo sin ser abrumados por el todo, o pueden mirar en las relaciones entre los módulos en puntos de vista que excluir el detalle del interior.
Los módulos de la capa de dominio deben surgir como una significativa parte del modelo, que cuenta la historia del dominio en una escala más grande
Los módulos permiten seguir con facilidad el concepto de 'acoplamientos débiles y alta cohesión'.
Cuando una aplicación comienza a ser compleja, es preferible dividirla en módulos.
Ejemplo:

- Ordenar por periodo, fecha, año fotos que se hallan guardado en un disco duro.
- División de los espacios de una casa (20 m^2, 30m^2, etc.)
- Las tablas de una base de datos.

Si un módulo se vuelve complejo durante el desarrollo, se debe refactorizar y convertir a su vez en varios módulos más pequeños.

7.9 ¿Qué son Agregados (Aggregates)?

Mientras los módulos suelen estar formados de clases relacionadas con los servicios o la funcionalidad de la aplicación, los agregados son grupos de entidades relacionadas entre sí a nivel de negocio.
Las entidades que están relacionadas entre sí y son dependientes entre ellas y se debe de poner en Agregados. En un agregado, se define cual va a ser la entidad raíz ("root") o principal y se da acceso público únicamente a esta. De modo que las entidades externas sólo pueden acceder a las entidades internas a través de la entidad raíz.
Ejemplo:

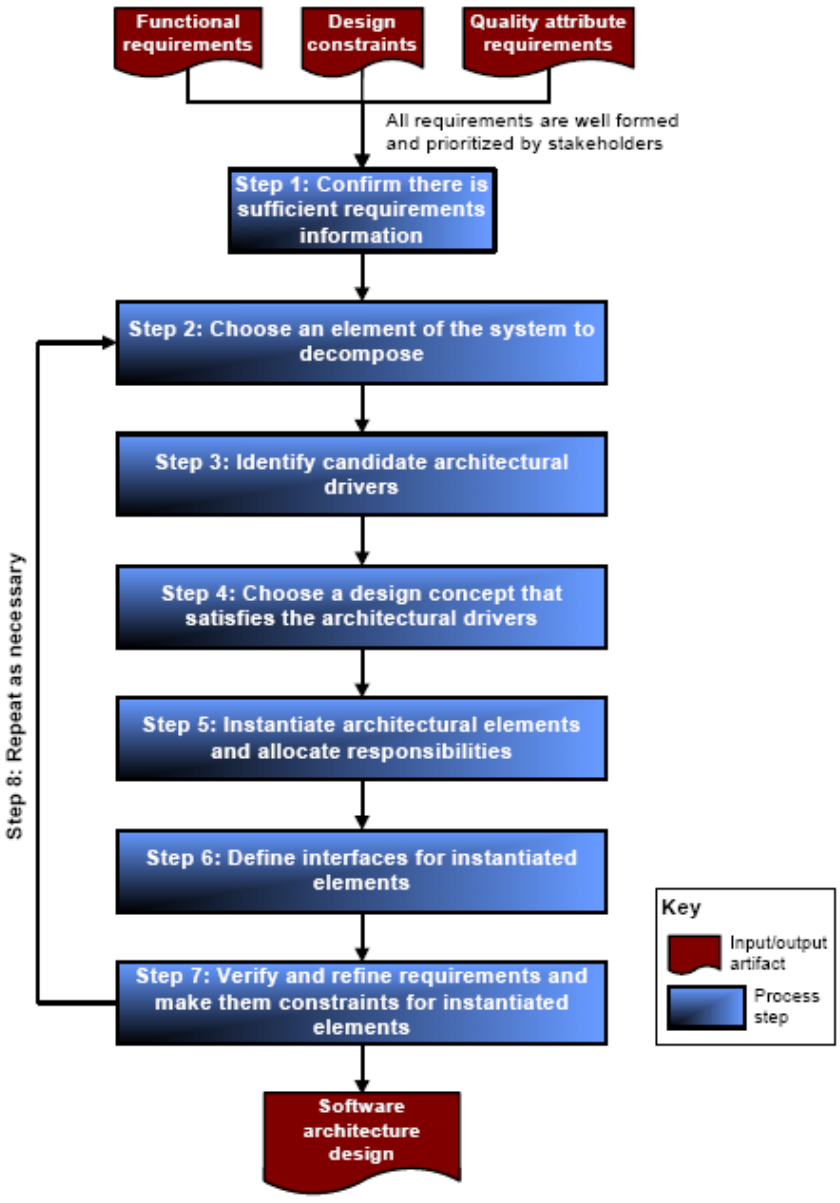
- Autorización para que mi amigo use mi pc.
- Pedir al bibliotecario un préstamo de un libro

7.10 ¿Qué son Factorías (Factories)?

Cuando la creación de una entidad o un agregado se convierte en un proceso complejo, delegamos esa responsabilidad en una Factoría. Las factorías son clases encargadas de crear entidades o agregados, construyendo todas las entidades contenidas en ellos. Son útiles especialmente cuando las reglas de creación de estas entidades o agregados son complejas.
Las Factorías permiten abstraer, separar la lógica y reglas de creación de una entidad y dejar en las entidades únicamente las reglas de negocio que son inherentes a ellas.
Es importante que el proceso de creación sea atómico y que se lance una excepción en el caso de que haya un error durante el proceso en lugar de devolver un objeto erróneo o incompleto.
Es posible que una Factoría utilice a su vez varias factorías para crear las entidades necesarias.
Por otro lado, hay que tener en cuenta que no es lo mismo crear un objeto desde cero que recuperarlo desde algún mecanismo de persistencia y reconstruirlo. En función de la implementación, este proceso puede realizarse un una única Factoría o en una para la creación y en otra para la reconstrucción y recuperación desde el mecanismo de persistencia.

8. Introducción a ADD (Attribute Driven Design)

8.1 ¿Qué es ADD?



Es una metodología para crear arquitecturas de software que tengan en cuenta los atributos de calidad del software. Previamente conocido como el método de diseño basado en la Arquitectura, pero debido a cuestiones de marca registrada el nombre fue cambiado por atributo de diseño impulsado hacia 2001. Esta metodología permite guiar al arquitecto en la realización de diseño arquitectónico de sistemas en equipos de trabajo pequeño, tomando como entrada los resultados obtenidos en el método QAW. Se puede decir, que una definición formal de ADD es: "Proceso de diseño basado en la descomposición iterativa del sistema. En cada iteración son seleccionados patrones y tácticas arquitectónicas con el fin de satisfacer los atributos de calidad".

8.2 Descripción del ciclo ADD

ADD sigue esencialmente el ciclo “Plan, Do and Check”:

Plan: Los atributos de calidad y las restricciones de diseño son consideradas para seleccionar qué tipos de elementos se utilizan en la arquitectura.

Do: Los elementos son instanciados para satisfacer los atributos de calidad, así como los requisitos funcionales.

Check: El resultado es analizado para determinar si los requisitos son satisfechos.

8.3 Entradas y Salidas del método ADD

Este método recibe como entrada una lista de drivers arquitecturales y produce a su salida una serie de estructuras que conforman al diseño de la arquitectura. Se va aplicando de forma iterativa. ADD es un método que sigue un enfoque de “divide y vencerás”; en la primera iteración del diseño, el elemento a descomponer es el sistema en sí. En iteraciones subsecuentes, el elemento a descomponer es un sub-elemento resultante de iteraciones previas. Generalmente se considera que el diseño de la arquitectura termina cuando se han tomado decisiones de diseño para satisfacer la mayor parte de los drivers en el tiempo permitido.

Ejemplo de entradas y salidas del método ADD

Entrada del Método ADD:

- Requerimientos funcionales
- Requerimientos de calidad
- Restricciones

Salida del Método ADD:

- Un conjunto de dibujos de vistas arquitectónicas, no una arquitectura detallada en toda regla.

8.4 Descomposición del sistema

El sistema es descompuesto por tres subsistemas. A continuación, explicaremos la consistencia de cada uno de ellos:

- **Subsistema Visor(#sVsor):**
  Este sistema permite capturar la información de un máximo de 100 metros a la redonda de proximidad de MetroBus a la estación en la cual se encuentra localizado el visor.

- **Subsistema Servidor(#sServ):**
  Este sistema permite capturar y enviar datos instantáneos dependiendo de la solicitud que realiza el tipo de cliente(se considera también el visor). También, soporta la APP por la cual los clientes pueden visualizar los buses más próximos a ellos con información detallada requerida y al sistema central de RasBus donde las computadoras de cada edificio que han recepcionado los datos del Bus por medio de un GPRS envían mediante el servidor los datos capturados con la espera de ser consultados por los clientes y visores de cada estación.

- **Subsistema GPRS(#sGprs):**
  Este sistema permite la captura de datos por medio del aparato de localización GPRS de cada bus que haya pasado por un punto específico mediante su recorrido y que será enviado al computador de cada edificio de RasBus.

Elemento	Motivantes Arquitectónicos Asociados
Subsistema Visor(#sVsor)	C-001 C-002 C-004 C-005
Subsistema Servidor(#sServ)	C-002 C-007 C-008 C-009
Subsistema GPRS(#sGprs)	C-003 C-006

- ✓ Celeste = Interfaz Hardware
- ✓ Morado = Interfaz Software

8.5 Interacción entre subsistemas

Los subsistemas interactúan por medio de canales de comunicación por donde se envía solicitudes e información dependiendo del tipo de entidad, ya sea Hardware o Software.

#sVsor >> #sServ

Interactúa con el #sServ en el envío de peticiones para:

- ✓ Localizar los buses en cualquier ubicación en un área de 100 metros a la redonda.
- ✓ Proveer un tiempo estimado de llegada de los buses a cada una de las paradas principales.
- ✓ Mostrar cinco buses más próximos en listado.

#sServ >> #sVsor

Interactúa con el #sVsor en el envío de peticiones para:

- ✓ Poder mostrar los datos necesarios requeridos por el visor.
- ✓ Habilitar o Deshabilitar al #sVsor en el envío de peticiones.

#sGprs >> #sServ

Interactúa con el #sGprs de manera que:

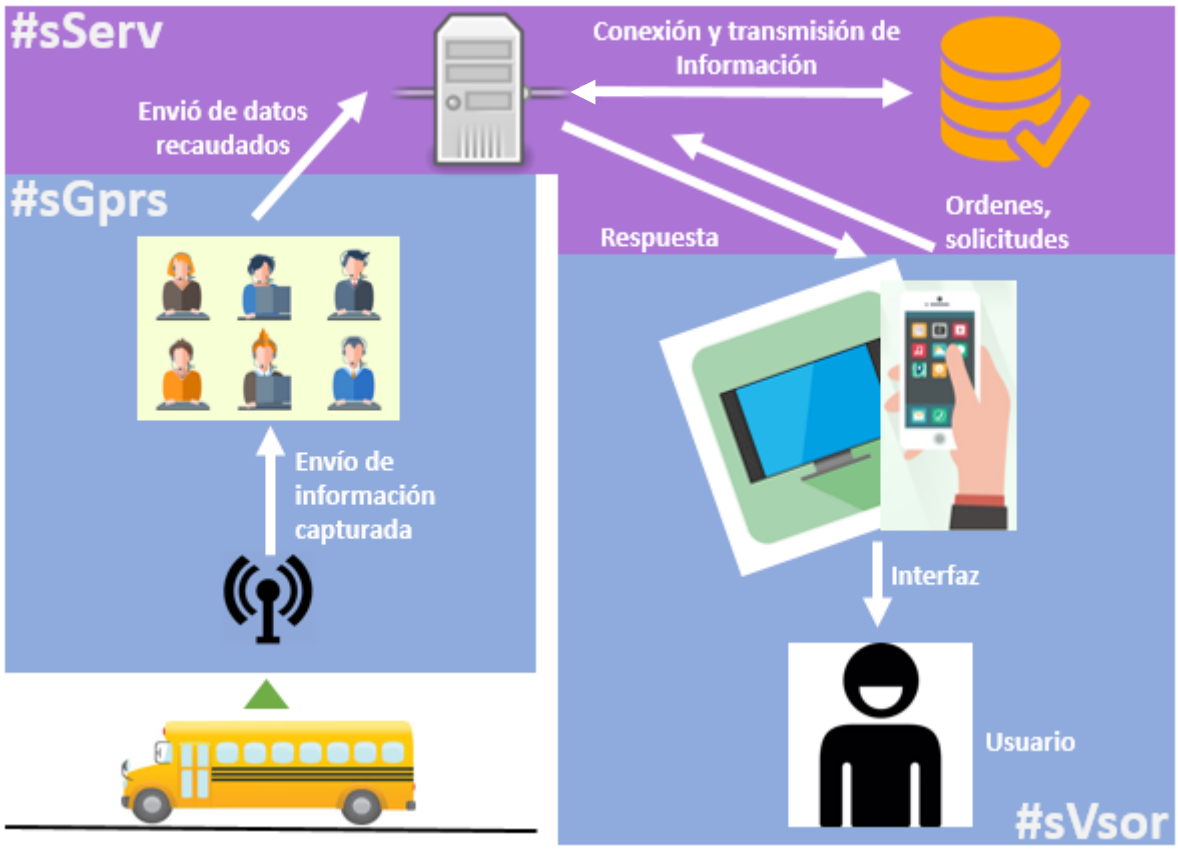
- ✓ Recibe información junto con el identificador del bus de manera regular.
- ✓ Envía vía internet los datos recaudados a las computadoras de cada edificio RasBus, que finalmente son llevados a través del servidor a la base de datos.

#sServ >> #sGprs

No interactúa con el #sGprs en el envió de peticiones, solo las recibe.

8.6 Diagrama de Contexto

De acuerdo a la información anterior que refiere el trabajo entre los subsistemas, a continuación, se presentara un diagrama de contexto que da una visión más detallada del sistema.



8.7 Iteración #2

Elegir un elemento del sistema a descomponer

Primero, el elemento a descomponer en esta iteración es el Subsistema Visor(#sVsor) ya que representa una mayor magnitud de relación con el usuario final.

Identificar candidatos a motivantes arquitectónicos principales

A continuación, se presenta los motivantes arquitectónicos obtenidos para este subsistema.

Motivantes Arquitectónicos	Valor de Importancia (1-10)
C-001	10
C-004	10
C-005	10

Identificar candidatos a motivantes arquitectónicos principales

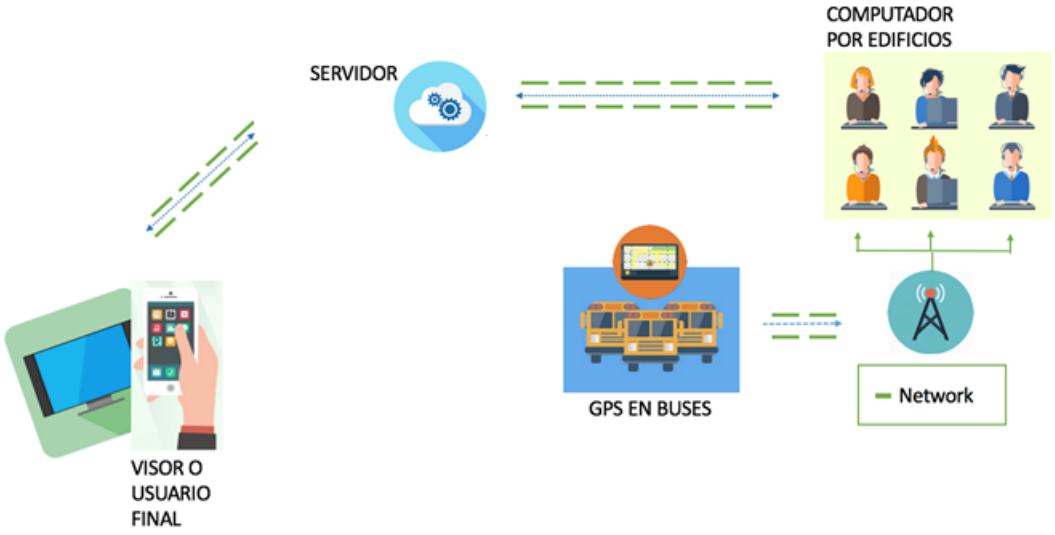
En este caso los motivantes arquitectónicos son:

Motivantes Arquitectónicos	Referencia	Criterio
C-001	**Sección 3.1**	Estos motivantes arquitectónicos describen la esencia de lo que los usuarios verán.
C-004		
C-005		

Especificación estructural del #sVsor

Este sistema permite capturar la información de un máximo de 100 metros a la redonda de proximidad de MetroBus a la estación en la cual se encuentra localizado el visor.

Modelo para el diseño del #sVsor



Línea 7B	Velocidad: 50 Km/h	Tiempo estimado de llegada: 3 min
Línea 1A	Velocidad: 30 Km/h	Tiempo estimado de llegada: 10 min
Línea 8C	Velocidad: 45 Km/h	Tiempo estimado de llegada: 12 min
Línea 9D	Velocidad: 37 Km/h	Tiempo estimado de llegada: 13 min
Línea 4D	Velocidad: 40 Km/h	Tiempo estimado de llegada: 15 min

8.8 Iteración #3

Elegir un elemento del sistema a descomponer

El elemento a descomponer en esta iteración es el Subsistema GPRS(#sGprs) que es un punto importante en el sistema para mantener localizados los Buses.

Identificar candidatos a motivantes arquitectónicos principales

A continuación, se presenta los motivantes arquitectónicos obtenidos para este subsistema.

Motivantes Arquitectónicos	Valor de Importancia (1-10)
C-003	10
C-006	10

Identificar candidatos a motivantes arquitectónicos principales

En este caso los motivantes arquitectónicos son:

Motivantes Arquitectónicos	Referencia	Criterio
C-003	**Sección 3.1**	Estos motivantes arquitectónicos describen la conexión que se tendrá mediante el GPRS
C-006		

Especificación estructural del #sGprs

Este sistema permite la captura de datos por medio del aparato de localización GPRS de cada bus que haya pasado por un punto específico mediante su recorrido y que será enviado al computador de cada edificio de RasBus. A continuación, se expondrá un gráfico de la composición del GPRS a usar:

8.9 Iteración #4

Elegir un elemento del sistema a descomponer

El último elemento a descomponer es el Subsistema Servidor(#sServ).

Identificar candidatos a motivantes arquitectónicos principales

los motivantes arquitectónicos son:

Motivantes Arquitectónicos	Valor de Importancia (1-10)
C-002	10
C-007	5
C-008	10
C-009	7

Los candidatos a motivantes arquitectónicos principales seleccionados para el #sServ ordenados de mayor a menor.

Motivantes Arquitectónicos	Referencia	Criterio
C-002	**Sección 3.1**	Estos motivantes arquitectónicos describen la esencia de lo que los usuarios del #sServ esperan obtener.
C-008		
C-009		
C-007		

Especificación estructural del #sServ

- El **#sServ** permite capturar y enviar datos instantáneos dependiendo de la solicitud que realiza el tipo de cliente. El #sServ soporta la APP por la cual los clientes pueden visualizar los buses más próximos a ellos con información detallada. Además, las computadoras de cada edificio por medio de un GPRS envían mediante el servidor los datos capturados con la espera de ser consultados por los clientes y visores de cada estación.

  Se plantea para el #sServ una estructura arquitectónica basada en el patrón "Modelo Vista Controlador", el cual permite como característica principal, facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento.

Requisitos específicos del sistema

A continuación, se definen sus requerimientos funcionales, atributos de calidad y restricciones de diseño que describen este subsistema(#sServ).

Requisitos Funcionales

- **Tiempo estimado de llegada**

Número de requisito	C-002
Nombre de requisito	Tiempo estimado de llegada
Fuente del requisito	#sServ
Prioridad del requisito	10

Descripción:

El sistema debe proveer un tiempo estimado de llegada de los buses a cada una de las paradas principales.

Entradas:

- Tiempo de cada bus.

Proceso:

- Calcula el tiempo de llegada de los buses hacia la parada

Salidas:

- Visualización de los buses

- **Datos históricos**

Número de requisito	C-007
Nombre de requisito	Datos históricos
Fuente del requisito	#sServ
Prioridad del requisito	5

Descripción:

El sistema debe permitir almacenar datos históricos de los tiempos de viajes por 24 meses.

Entradas:

- Datos del viaje.

Proceso:

- Se captura la información de los buses (tiempo de llegada, hora de salida, etc.) y de inserta en la tabla correspondiente en la base de datos.

Salidas

- Histórico de buses.

- **Geolocalización de Paraderos**

Número de requisito	C-008
Nombre de requisito	Geolocalización de paraderos
Fuente del requisito	#sServ
Prioridad del requisito	10

Descripción:

El cliente puede revisar la información de RasBus en un aplicativo móvil que le indicará que paradero está más cerca según su ubicación y a qué hora llega el bus más próximo.

Entradas:

- Datos del paradero más cercano.

Proceso:

- Mediante la ubicación del dispositivo móvil, se le da la posición del paradero más cercano al usuario.

Salidas

- Vista del Paradero.

- **Algoritmo de Estimación**

Número de requisito	C-009
Nombre de requisito	Algoritmo de Estimación
Fuente del requisito	#sServ
Prioridad del requisito	7

Descripción:

El sistema requiere las entradas al algoritmo de estimación, lo cual tiene implicaciones de almacenamiento (posición y velocidad); como también implicaciones de ancho de banda de comunicación.

Entradas:

- Datos del servidor.

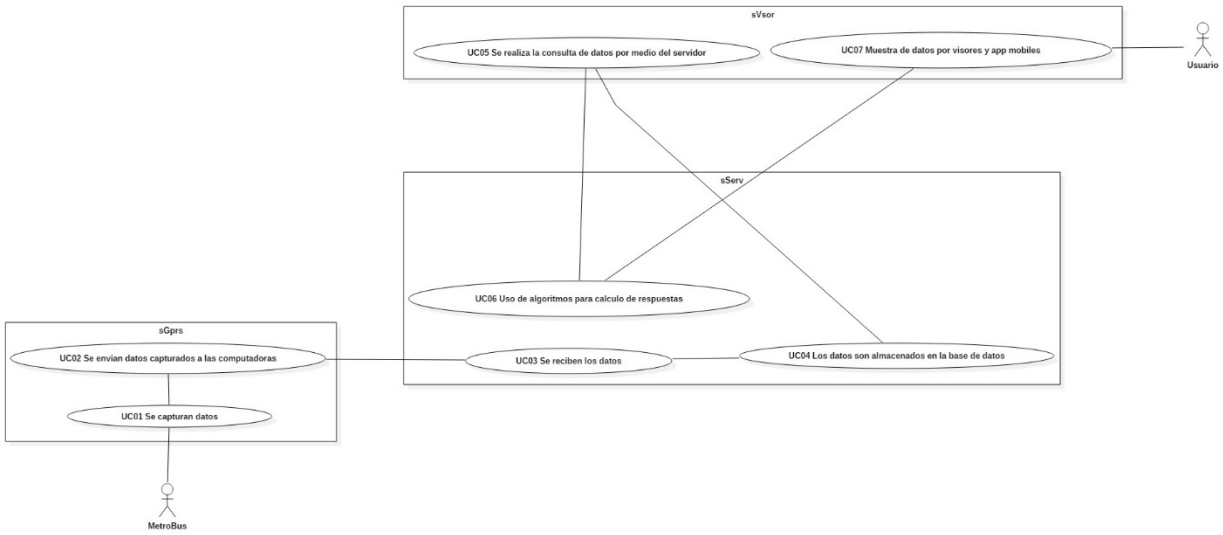
Proceso:

- Se captura la siguiente información (posición, velocidad y ancho de banda) mediante algoritmos.

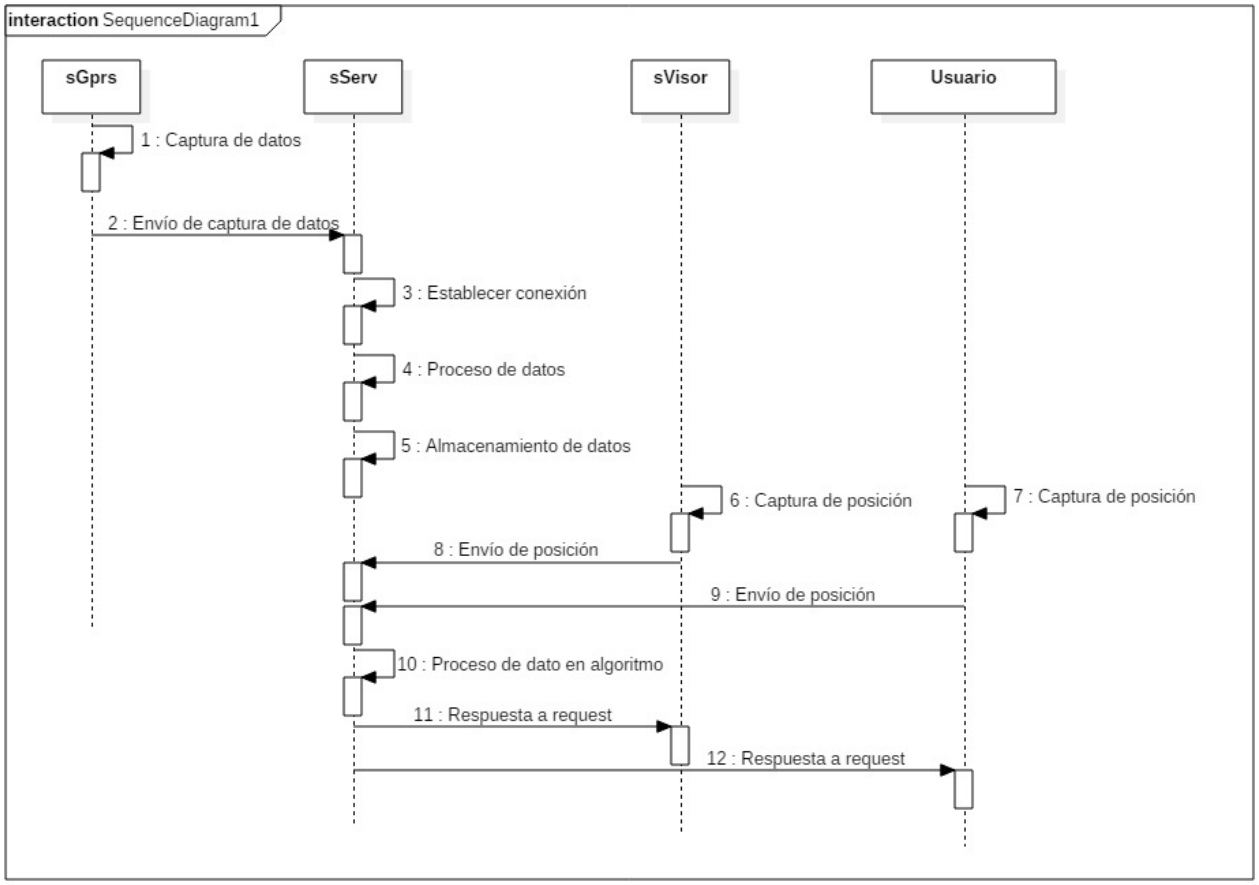
Salidas

- Datos obtenidos para el servidor.

9. Diagrama de caso de uso del Sistema



10. Diagrama de secuencia del Sistema



11. Resultados de ADD

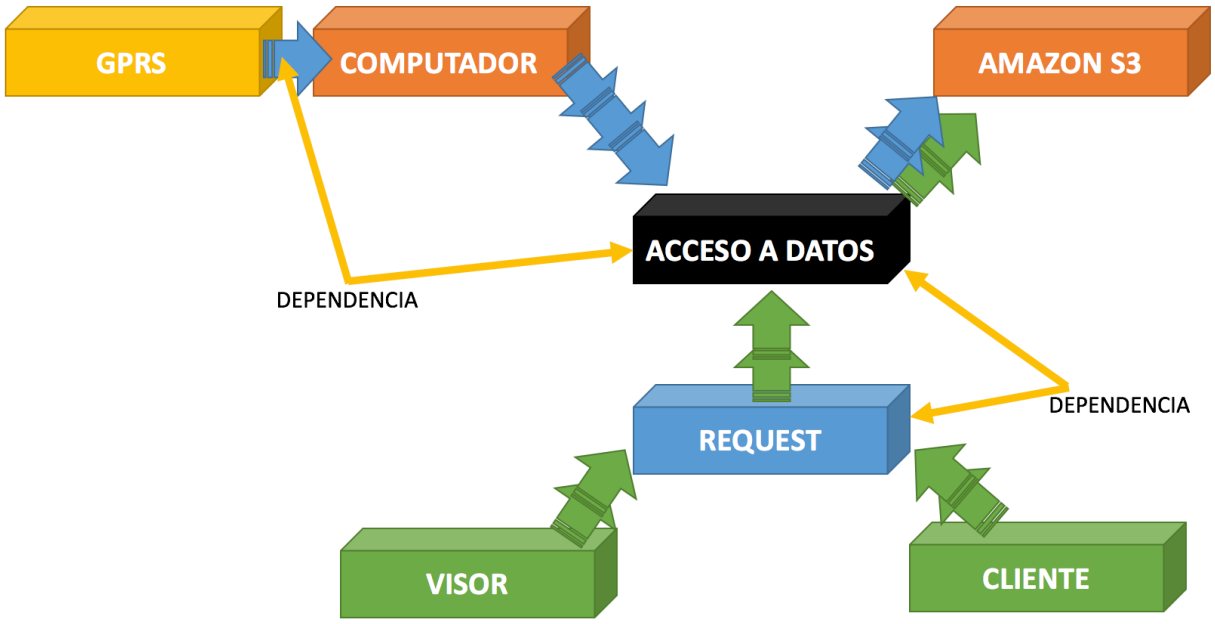
La finalidad de hacer uso del método ADD es mantener una descripción modular del sistema, descomponiéndolo incluso hasta llegar a los componentes primarios del mismo.

A continuación, se darán a conocer los resultados de cada iteración:

Driver	Patrón	Comentarios
1	Patrón de Identidades de múltiples servicios confiables	El servidor debe contar con dos roles quienes tienen diferentes permisos en la base de datos.
2	Cliente / Servidor	La conexión se realiza entre la capa cliente y la capa servidor.
3	Persistencia	Datos persistentes almacenados en la base de datos que serán leídos por otros procesos a lo largo de vida del sistema.
4	SOA PIPELINES	Brindar servicios en respuesta a una petición del consumidor. Servicios sin estado con facilidad de ser orquestadas.

12. Vista Modular

La solución propuesta requiere el uso de un estilo de arquitectura de software REST & SOA. A continuación, se expone la vista modular desde un punto de vista abstracto del proyecto:



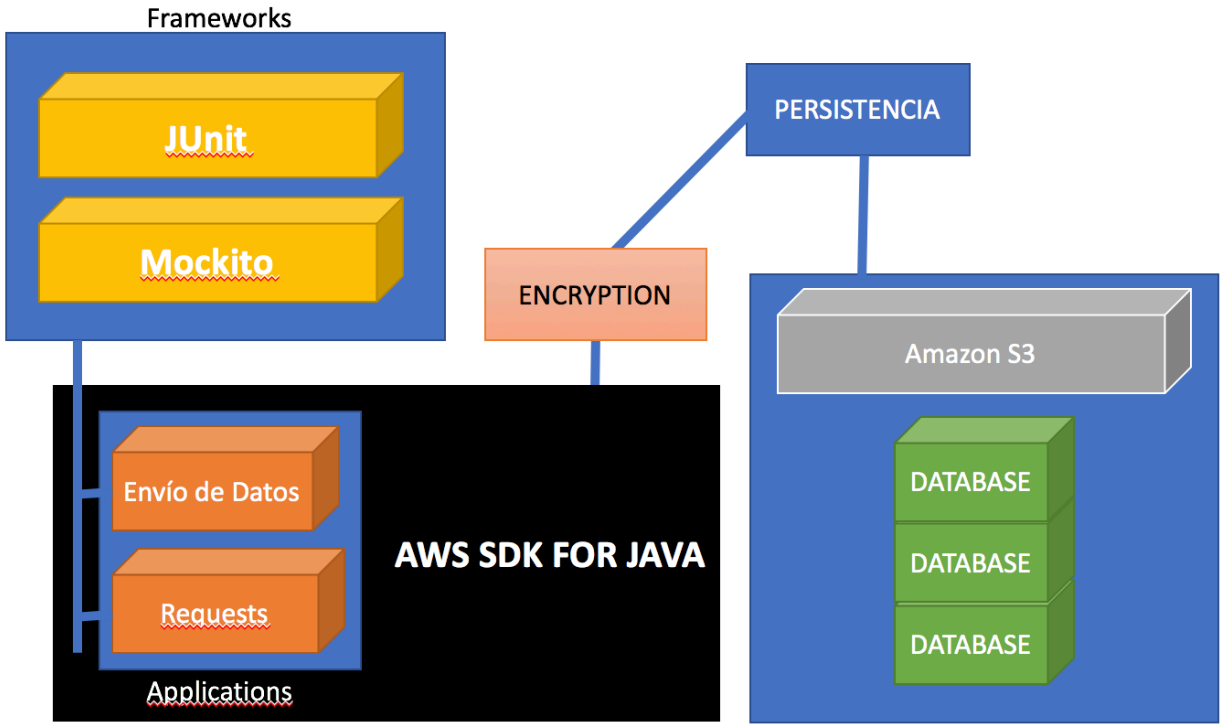
GPRS y Computador: Estos dos módulos constituyen un componente esencial en el marco de trabajo del proyecto al cual nos referiremos como *MODULO DE CONEXIÓN PRINCIPAL.* En este módulo se realiza el envío de datos capturados por el GPRS a los computadores localizados en la torre principal de Rasbus.

Módulo de Servicio: El acceso a datos es un servicio que permite realizar consultas y envío de datos mediante el servidor al gestor de la base de datos. Como se puede apreciar en la gráfica, contamos con dos actores muy importantes para la obtención y manejo de datos como lo son el VISOR y EL CLIENTE (APP MOBILE), donde se generan nodos de dependencias para el buen desempeño del sistema. Es decir, si no hay datos de captura por el GRPS el sistema no tomaría el servicio de acceso a datos y por lo tanto el sistema estaría en modo OFF. De igual manera, tomemos como ejemplo el visor, este tiene un receptor que permite tomar la localización del mismo y mediante algoritmos obtener los buses más próximos y tiempo estimado de llegada, pero si este no hiciera consulta alguna el sistema cumpliría la única función de almacén de datos.

Amazon S3: Este es un servicio de almacén de datos y archivos brindado por Amazon donde también es posible hacer un repositorio del proyecto, brinda facilidades de migración y adaptable a web servers más usados hoy en día.

13. Vista Runtime

Para la siguiente solución, la aplicación Web contendrá información recibida de los ordenadores de la torre principal Rasbus y mantendrá los servicios web a los clientes. Dado que el sistema solo requiere almacenar datos y ser consultados es que se expone el siguiente modelo de vista Runtime:



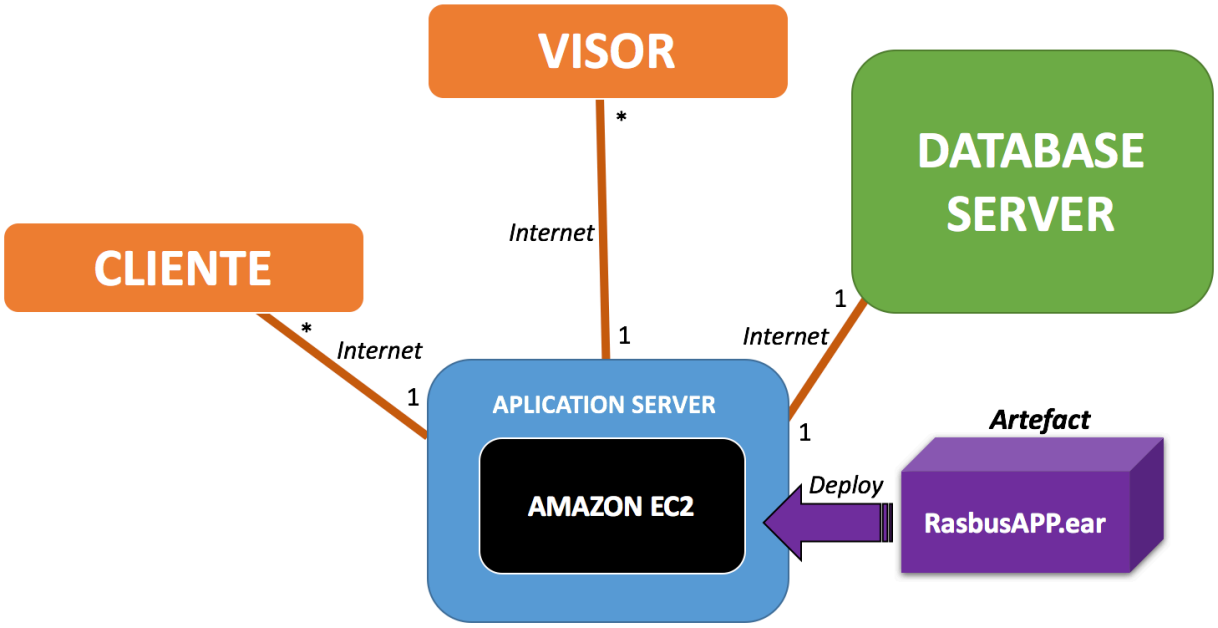
Detalle de componentes:

Frameworks: Para hacer uso de Amazon S3 como gestor de base de datos se requieren los frameworks JUnit & Mockito, además de AWS SDK FOR JAVA. Este toolkit proporciona API de JAVA para el sencillo manejo de servicios AMAZON WEB SERVICES.

Servicios Web: Estos son los principales servicios que deben lograr satisfacer las funcionalidades esenciales del sistema. En este servicio se realiza la encriptación de datos y transacción con la base de datos.

14. Vista Despliegue

Debido a la solución propuesta se expone la siguiente vista de despliegue:



Aplication Server: Se hará uso de Amazon EC2 que cumple la función de brindar hospedaje de servidores virtuales (HOSTING) donde será cargado el sistema en su totalidad. De este manera ya integrado el servidor de base de datos AMAZON S3, el sistema está listo para ser operado por los clientes y visores.

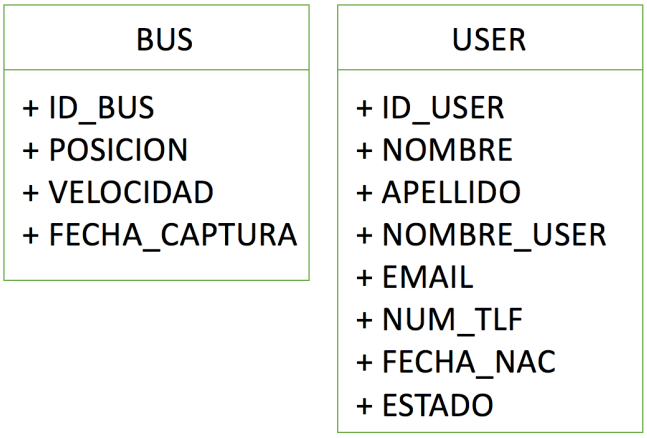
15. Vista Modelo de Dato

Debido a los requerimientos capturados en el proyecto, la base de datos consiste en dos entidades tales como:

- Bus, donde tiene como atributos información necesaria para por medio de algoritmos realizar los procesos de respuesta esperados por las peticiones del cliente o visor.

- User, para gestionar las sesiones por medio de aplicaciones móviles.

A continuación, se presenta una gráfica de las tablas anteriormente mencionadas:



16. Sustento de eleccion web server y base de datos

[4]***Título de fuente***: MySQL vs PostgreSQL vs MongoDB (velocidad)

[5]***Título de fuente***: What are the advantage of using MongoDB GridFS vs Amazon S3 to store assets for a product with MongoDB as the database backend?
[6]***Título de fuente***: Microsoft Azure vs. Amazon Web Services: Cloud Comparison

[7]***Título de fuente***: Google Cloud vs AWS: a comparison

[4] http://macool.me/mysql-vs-postgresql-vs-mongodb-velocidad/04
[5] https://www.quora.com/What-are-the-advantage-of-using-MongoDB-GridFS-vs-Amazon-S3-to-store-assets-for-a-product-with-MongoDB-as-the-database-backend
[6] http://www.tomsitpro.com/articles/azure-vs-aws-cloud-comparison,2-870.html
[7] http://cloudacademy.com/blog/google-cloud-vs-aws-a-comparison/