FUNDAMENTOS DE LA PROGRAMACIÓN

CONSTRUCCIÓN DE SOFTWARE

LA ACTIVIDAD DE CONSTRUCCIÓN DE SOFTWARE ABARCA TAREAS DE CODIFICACIÓN Y LA REALIZACIÓN DE PRUEBAS, QUE CONDUCEN A UN SOFTWARE OPERATIVO QUE ESTÁ LISTO PARA ENTREGARLO AL CLIENTE O USUARIO FINAL.

CODIFICACIÓN, EN INGENIERÍA DEL SOFTWARE MODERNA, PUEDE SER:

- **LA CREACIÓN DIRECTA DE CÓDIGO FUENTE DE UN LENGUAJE DE PROGRAMACIÓN.**
- **LA GENERACIÓN AUTOMÁTICA DE CÓDIGO FUENTE, AL UTILIZAR UNA REPRESENTACIÓN INTERMEDIA DEL DISEÑO DEL COMPONENTE QUE SERÁ CONSTRUIDO.**
- **LA GENERACIÓN AUTOMÁTICA DE CÓDIGO, MEDIANTE UN LENGUAJE DE PROGRAMACIÓN DE CUARTA GENERACIÓN.**

FUNDAMENTOS DE LA PROGRAMACIÓN

CONSTRUCCIÓN DE SOFTWARE

LAS PRUEBAS SE REALIZAN EN DISTINTOS NIVELES:

- **PRUEBAS DE UNIDAD: COMPONENTES**
- **PRUEBAS DE INTEGRACIÓN: REALIZADAS MIENTRAS EL SISTEMA ESTÁ EN CONSTRUCCIÓN.**
- **PRUEBAS DE VALIDACIÓN: EVALÚAN SI LOS REQUISITOS HAN SIDO SATISFECHOS PARA EL SISTEMA COMPLETO.**
- **PRUEBAS DE ACEPTACIÓN: LO REALIZA EL CLIENTE EN UN ESFUERZO ENCAMINADO A EJERCITAR LAS CARACTERÍSTICAS Y FUNCIONES.**

FUNDAMENTOS DE LA PROGRAMACIÓN

CONSTRUCCIÓN DE SOFTWARE

EXISTE UNA SERIE DE PRINCIPIOS Y CONCEPTOS APLICABLES A LA CODIFICACIÓN Y LAS PRUEBAS:

PRINCIPIOS Y CONCEPTOS DE CODIFICACIÓN

LOS PRINCIPIOS Y CONCEPTOS QUE GUÍAN LA TAREA DE CODIFICACIÓN ESTÁN ALINEADOS DE MANERA MUY CERCANA AL ESTILO DE LA PROGRAMACIÓN, LOS LENGUAJES DE LA PROGRAMACIÓN Y LOS MÉTODOS DE PROGRAMACIÓN.

- **PRINCIPIOS DE PREPARACIÓN**
- **PRINCIPIOS DE CODIFICACIÓN**
- **PRINCIPIOS DE VALIDACIÓN**

FUNDAMENTOS DE LA PROGRAMACIÓN

CONSTRUCCIÓN DE SOFTWARE

PRINCIPIOS DE PREPARACIÓN: ANTES DE ESCRIBIR UNA LÍNEA DE CÓDIGO SE DEBE ESTAR SEGURO DE:

- **ENTENDER EL PROBLEMA QUE SE INTENTA RESOLVER.**
- **ENTENDER LOS PRINCIPIOS Y CONCEPTOS BÁSICOS DEL DISEÑO.**
- **ESCOGER UN LENGUAJE DE PROGRAMACIÓN QUE SATISFAGA LAS NECESIDADES DEL SOFTWARE QUE SE VA A CONSTRUIR Y EL AMBIENTE EN EL QUE ÉSTE VA A OPERAR.**
- **SELECCIONAR UN AMBIENTE DE PROGRAMACIÓN QUE PROPORCIONE HERRAMIENTAS QUE FACILITEN EL TRABAJO.**
- **CREAR UN CONJUNTO DE PRUEBAS DE UNIDAD QUE SERÁN APLICADAS UNA VEZ QUE SE COMPLETE EL COMPONENTE QUE SE VA A CODIFICAR.**

FUNDAMENTOS DE LA PROGRAMACIÓN

CONSTRUCCIÓN DE SOFTWARE

PRINCIPIOS DE CODIFICACIÓN:

- **RESTRINGIR LOS ALGORITMOS AL SEGUIR LA PRÁCTICA DE LA PROGRAMACIÓN ESTRUCTURADA.**
- **SELECCIONAR LAS ESTRUCTURAS DE DATOS SEGÚN LAS NECESIDADES DEL DISEÑO.**
- **ENTENDER LA ARQUITECTURA DEL SOFTWARE Y CREAR INTERFACES QUE SEAN CONSISTENTES CON ELLA.**
- **MANTENER LA LÓGICA CONDICIONAL TAN SIMPLE COMO SEA POSIBLE.**
- **CREAR CICLOS ANIDADOS EN FORMA QUE LOS HAGA FÁCILES DE PROBAR.**
- **SELECCCIONAR NOMBRES DE VARIABLES SIGNIFICATIVAS Y SEGUIR ESTÁNDARES LOCALES DE CODIFICACIÓN.**
- **ESCRIBIR CÓDIGO QUE TENGA DOCUMENTACIÓN PROPIA.**
- **CREAR UNA CONFIGURACIÓN LINEAL, POR EJEMPLO: SANGRÍAS Y LÍNEAS EN BLANCO QUE AYUDEN A LA COMPRENSIÓN DEL CÓDIGO.**

FUNDAMENTOS DE LA PROGRAMACIÓN

CONSTRUCCIÓN DE SOFTWARE

PRINCIPIOS DE VALIDACIÓN: DESPUES DE HABER COMPLETADO LOS PRIMEROS PASOS DE CODIFICACIÓN, SE DEBE ESTAR SEGURO DE:

- **CONDUCIR UN ENSAYO DE CÓDIGO CUANDO SEA APROPIADO.**
- **REALIZAR PRUEBAS DE UNIDAD Y CORREGIR LOS ERRORES QUE SE HAYAN DESCUBIERTO.**
- **REFABRICAR EL CÓDIGO.**

FUNDAMENTOS DE LA PROGRAMACIÓN

CONSTRUCCIÓN DE SOFTWARE

PRINCIPIOS DE LAS PRUEBAS

GLEN MYERS [MYE79] ESTABLECE UNA SERIE DE REGLAS QUE PUEDEN SERVIR COMO OBJETIVOS DE LAS PRUEBAS:

- **LAS PRUEBAS CONSISTEN EN UN PROCESO EN EL QUE SE EJECUTA UN PROGRAMA CON LA INTENCIÓN DE ENCONTRAR UN ERROR QUE AÚN NO SE DESCUBRE.**
- **UN BUEN CASO DE PRUEBA ES AQUEL EN EL QUE HAY UNA GRAN PROBABILIDAD DE ENCONTRAR UN ERROR .**
- **UNA PRUEBA EXITOSA ES AQUELLA QUE ENCUENTRA UN ERROR QUE AÚN NO SE DESCUBRE.**

FUNDAMENTOS DE LA PROGRAMACIÓN

CONSTRUCCIÓN DE SOFTWARE

DAVIS [DAV95] SUGIERE UN CONJUNTO DE PRINCIPIOS PARA LAS PRUEBAS:

- **LAS PRUEBAS SE DEBEN PLANEAR MUCHO ANTES DE QUE COMIENCE EL PROCESO DE PRUEBA.**
- **LAS PRUEBAS DEBEN COMENZAR "EN LO PEQUEÑO" Y PROGRESAR HACIA "LO GRANDE".**
- **LAS PRUEBAS EXHAUSTIVAS NO SON POSIBLES.**

AUNQUE EXISTEN MUCHOS PRINCIPIOS PARA LAS PRUEBAS, SÓLO UNO ES EL DOMINANTE:

LAS PRUEBAS SE FORMAN CON UN PROCESO EN EL QUE UN PROGRAMA SE EJECUTA CON EL OBJETIVO DE ENCONTRAR ERRORES .

FUNDAMENTOS DE LA PROGRAMACIÓN

VERIFICACIÓN Y VALIDACIÓN

LA PRUEBA ES UN ELEMENTO DE UN TEMA MÁS AMPLIO QUE SUELE DE NOMINARSE *VERIFICACIÓN* Y *VALIDACIÓN*.

- **VERIFICACIÓN: ES EL CONJUNTO DE ACTIVIDADES QUE ASEGURAN QUE EL SOFTWARE IMPLEMENTE CORRECTAMENTE UNA FUNCIÓN ESPECÍFICA.**
- **VALIDACIÓN: ES UN CONJUNTO DIFERENTE DE ACTIVIDADES QUE ASEGURAN QUE EL SOFTWARE CONSTRUIDO CORRESPONDE CON LOS REQUISITOS DEL CLIENTE**

VERIFICACIÓN: ¿ESTAMOS CONSTRUYENDO EL PRODUCTO CORRECTAMENTE?

VALIDACIÓN: ¿ESTAMOS CONSTRUYENDO EL PRODUCTO CORRECTO?

FUNDAMENTOS DE LA PROGRAMACIÓN

VERIFICACIÓN Y VALIDACIÓN

LAS ETAPAS DE *VERIFICACIÓN* Y *VALIDACIÓN* ABARCAN MUCHAS ACTIVIDADES RELACIONADAS CON LA *CALIDAD DEL SOFTWARE*; Y COMPRENDE UNA AMPLIA LISTA DE ACTIVIDADES:

- **REVISONES TÉCNICAS FORMALES.**
- **AUDITORIAS DE CALIDAD Y DE CONFIGURACIÓN.**
- **MONITOREO DEL DESEMPEÑO.**
- **SIMULACIÓN.**
- **FACTIBILIDAD.**
- **REVISIÓN DE LA DOCUMENTACIÓN Y LA BASE DE DATOS.**
- **ANÁLISIS DE ALGORITMOS.**
- **PRUEBAS DE DESARROLLO, DE FACILIDAD DE USO, CALIFICACIÓN Y DE INSTALACIÓN**

FUNDAMENTOS DE LA PROGRAMACIÓN

VERIFICACIÓN Y VALIDACIÓN

LA *CALIDAD* SE INCORPORA AL SOFTWARE EN TODO EL PROCESO DE INGENIERÍA

LA APLICACIÓN APROPIADA DE MÉTODOS Y HERRAMIENTAS, LAS REVISIONES TÉCNICAS FORMALES Y EFECTIVAS, JUNTO CON UNA ADMINISTRACIÓN Y UNA MEDICIÓN SÓLIDAS APORTAN LA CALIDAD, QUE SE CONFIRMA DURANTE LAS PRUEBAS.

MILLER, RELACIONA LA PRUEBA DEL SOFTWARE CON LA CALIDAD AL AFIRMAR:

LO QUE MOTIVA LA PRUEBA DE LOS PROGRAMAS ES LA CONFIRMACIÓN DE LA CALIDAD DEL SOFTWARE CON MÉTODOS QUE SE PUEDAN APLICAR DE MANERA ECONÓMICA Y EFECTIVA EN SISTEMAS GRANDES Y PEQUEÑOS.

FUNDAMENTOS DE LA PROGRAMACIÓN

ORGANIZACIÓN PARA LAS PRUEBAS DEL SOFTWARE

EL DESARROLLADOR DE SOFTWARE SIEMPRE SERÁ EL RESPONSABLE DE:

- **PROBAR LAS UNIDADES INDIVIDUALES (COMPONENTES) DEL PROGRAMA Y ASEGURAR QUE CADA UNA REALICE LA FUNCIÓN O MUESTRE EL COMPORTAMIENTO PARA EL QUE SE DISEÑÓ.**

- **EN MUCHOS CASOS, EL DESARROLLADOR TAMBIÉN APLICA LA PRUEBA DE INTEGRACIÓN (CONSTRUCCIÓN Y PRUEBA DE TODA LA ARQUITECTURA DEL SOFTWARE).**

FUNDAMENTOS DE LA PROGRAMACIÓN

ORGANIZACIÓN PARA LAS PRUEBAS DEL SOFTWARE

SÓLO DESPUÉS DE QUE LA ARQUITECTURA DEL SOFTWARE ESTÉ COMPLETA PARTICIPARÁ UN GRUPO INDEPENDIENTE DE PRUEBA.

EL DESARROLLADOR DEL SOFTWARE Y EL GRUPO INDEPENDIENTE DE PRUEBA DEBEN TRABAJAR UNIDOS EN TODO EL PROYECTO DE SOFTWARE PARA ASEGURAR PRUEBAS EXHAUSTIVAS.

CUANDO LAS PRUEBAS SE REALIZAN, EL DESARROLLADOR DEBE ESTAR DISPONIBLE PARA CORREGIR LOS ERRORES QUE SE DESCUBREN.

EL GRUPO INDEPENDIENTE DE PRUEBA ES PARTE DEL EQUIPO DEL PROYECTO DE DESARROLLO DEL SOFTWARE, PARTICIPA EN EL ANÁLISIS , DISEÑO Y EN TODOS LOS PASOS DE UN PROYECTO GRANDE, PLANEA Y ESPECIFICA PROCEDIMIENTOS DE PRUEBA.

FUNDAMENTOS DE LA PROGRAMACIÓN

TÉCNICAS DE PRUEBA DE SOFTWARE

PARA CUALQUIER PRODUCTO DE INGENIERÍA EXISTEN DOS ENFOQUES A LA HORA DE PROBAR UN PRODUCTO:

PRUEBAS DE CAJA BLANCA:

Se centra en el estudio minucioso de la operatividad de una parte del sistema considerando los detalles procedurales o sea la Lógica del sistema.

PRUEBAS DE CAJA NEGRA:

Analiza principalmente la compatibilidad entre sí, en cuanto a las interfaces, de cada uno de los componentes del software (no tiene en cuenta la lógica del sistema).

ENFOQUES:

- **Caja Blanca (como lo hace)**
- **Caja Negra (que es lo que hace)**

COMBINANDO AMBAS ESTRATEGIAS

SE OBTIENE UNA COMPLETA

VALIDACIÓN DEL SOFTWARE

FUNDAMENTOS DE LA PROGRAMACIÓN

PRUEBA DE SOFTWARE – PRUEBAS DE CAJA NEGRA

Las pruebas de caja negra se llevan a cabo sobre la interfaz del software, sin considerar el comportamiento interno y la estructura del programa.

LOS CASOS DE PRUEBA DE LA CAJA NEGRA PRETENDEN DEMOSTRAR QUE:

- **Las funciones del software son operativas.**
- **La entrada se acepta de forma correcta.**
- **Se produce una salida correcta.**
- **La integridad de la información externa se mantiene.**

LAS PRUEBAS DE CAJA NEGRA PRETENDEN ENCONTRAR LOS SIGUIENTES TIPOS DE ERRORES:

- **Funciones incorrectas o ausentes.**
- **Errores en la interfaz.**
- **Errores en estructuras de datos o en accesos a bases de datos externas.**
- **Errores de rendimiento.**
- **Errores de inicialización y de terminación.**

EXISTEN DOS TECNICAS DE PRUEBA DE CAJA NEGRA:

- **Técnica de Partición de Equivalencia.**
- **Técnica de Análisis de Valores Límites.**

FUNDAMENTOS DE LA PROGRAMACIÓN

PRUEBAS DE CAJA NEGRA – TÉCNICA DE PARTICIÓN DE EQUIVALENCIA

Este método de prueba de caja negra divide el dominio de entrada de un programa en clases de datos, a partir de las cuales derivan los casos de prueba.

Cada una de estas clases de equivalencia representa a un conjunto de estados válidos o inválidos para las condiciones de entrada.

1- IDENTIFICACIÓN DE LAS CLASES DE EQUIVALENCIA

Se identifican clases de equivalencia válidas e inválidas con una tabla

CONDICIONES EXTERNAS	CLASES DE EQUIVALENCIA VÁLIDAS	CLASES DE EQUIVALENCIA INVÁLIDAS

TENIENDO EN CUENTA LAS SIGUIENTES DIRECTRICES:

Si una condición de entrada especifica un rango de valores (entre 1 y 999), se define:

una CEV (1 <= valor <= 999) y dos CEI valor < 1 y valor > 999.

Si una condición requiere un valor específico (el primer carácter tiene que ser una letra), se define:

una CEV (una letra) y una CEI (no es una letra).

Si una condición especifica un conjunto de valores de entrada, se define:

una CEV para cada uno de los valores válidos (CEV para "Moto", "Coche" y "Camión"), y una CEI <> de "Moto", "Coche" y "Camión"

FUNDAMENTOS DE LA PROGRAMACIÓN

PRUEBAS DE CAJA NEGRA – PARTICIÓN DE EQUIVALENCIA

2. IDENTIFICACIÓN DE CASOS DE PRUEBA

- **Asignar un número único a cada clase de equivalencia.**
- **Escribir casos de prueba hasta que sean cubiertas todas las CEV, intentando cubrir en cada caso tantas CEV como sea posible.**
- **Para cada CEI, escribir un caso de prueba, cubriendo en cada caso una CEI.**

EJEMPLO

Diseñar casos de prueba utilizando la técnica de Partición de Equivalencia para un módulo que ingrese los datos de entrada para un Registro de películas, si son correctos los almacena sino informar el Error.

- **Título: alfanumérico**
- **Año estreno: de 1990 a 2021**
- **Tipo película: Comedia, Drama, Terror**

CONDICIONES EXTERNAS	CLASES DE EQUIVALENCIA VÁLIDAS	CLASES DE EQUIVALENCIA INVÁLIDAS
TÍTULO (T)	T = comienza con letra	T = NO comienza de letra
AÑO DE ESTRENO (A)	1990 >= A < = 2021	A < 1990 A > 2021
TIPO PELÍCULA (P)	P= comedia P = drama P = terror	P <> comedia/ drama / terror

FUNDAMENTOS DE LA PROGRAMACIÓN

CONDICIONES EXTERNAS	CLASES DE EQUIVALENCIA VÁLIDAS	CLASES DE EQUIVALENCIA INVÁLIDAS
TÍTULO (T)	1 T = comienza con letra	6 T < > de letra
AÑO DE ESTRENO (A)	2 - 1990 < = A < = 2021	7 - A < 1990 8 - A > 2021
TIPO PELÍCULA (P)	3 - P= comedia 4 - P = drama 5 - P = terror	9 - P < > comedia/ drama / terror

Tabla con Casos de Prueba

T	A	P	CONDICIONES	RESULTADOS
Mujer Bonita	2005	comedia	1, 2, 3	válido
Titanic	1998	drama	1, 2, 4	válido
Dracula	2010	terror	1, 2, 5	Válido
54TT	2007	drama	6, 2, 4	Inválido T
Romance	1987	comedia	1, 7, 3	Inválido A
wwwwijiwjwj	2022	terror	1, 8, 5	Inválido A
Car	1984	infantil	1, 2, 9	Inválido P

FUNDAMENTOS DE LA PROGRAMACIÓN

PRUEBAS DE CAJA NEGRA – TÉCNICA DE ANÁLISIS DE VALORES LÍMITES

La técnica de Análisis de Valores Límites (AVL), se basa en la EVIDENCIA EXPERIMENTAL de que los ERRORES suelen aparecer con MAYOR PROBABILIDAD en los EXTREMOS DE LOS CAMPOS DE ENTRADA.

Es una variante y refinamiento de la Partición de Equivalencia con dos diferencias principales:

- **En lugar de seleccionar cualquier elemento en una clase de equivalencia, como representativo, los elementos son seleccionados de manera que cada extremo de la clase de equivalencia sea probada.**

- **En vez de concentrarse exclusivamente en condiciones de entrada, se exploran las condiciones de salida definiendo las clases de equivalencia de salida.**

FUNDAMENTOS DE LA PROGRAMACIÓN

PRUEBAS DE CAJA NEGRA – ANÁLISIS DE VALORES LÍMITES

Las directrices para el Análisis de Valores Límites son muy similares a las proporcionadas para Partición de Equivalencia:

- **Si una condición de entrada especifica un rango limitado por los valores a inferior y b superior, los casos de prueba deben diseñarse con esos valores, además de los que se encuentran por encima y por debajo de ellos.**
- **Si una condición requiere un valor específico a, se diseña un caso de prueba para ese valor, además de los que se encuentran por encima y por debajo de ellos.**
- **Si una condición especifica un conjunto de valores de entrada, se diseña un caso de prueba para cada elemento y uno que no pertenezca al conjunto.**

FUNDAMENTOS DE LA PROGRAMACIÓN

PRUEBAS DE CAJA NEGRA – ANÁLISIS DE VALORES LÍMITES

EJEMPLO: Se desean diseñar los casos de pruebas utilizando la técnica de caja negra AVL sobre un módulo utilizado por una empresa de transporte, el cual desea calcular la tarifa a pagar según el trayecto, la cantidad de pasajes (hasta 60) y la edad del pasajero. Dicha empresa sólo realiza viajes hacia Tucumán y Córdoba.

DATOS DE ENTRADA:

- **CIUDAD DESTINO (CD): campo que puede tomar los mismos valores "TUC" y "CBA".**
- **CANTIDAD DE PASAJES: (CP) indica la cantidad de pasajes comprados por una persona.**
- **EDAD: es un campo numérico positivo.**

La tarifa a cobrar además de estar en función del trayecto realizado, ofrece los siguientes descuentos por cantidad de pasajes solicitados y edad del pasajero.

- **15% de descuento si la cantidad de pasaje es superior a 4 e inferior a 8.**
- **40% a los pasajeros con edad igual a 65 años.**

FUNDAMENTOS DE LA PROGRAMACIÓN

PRUEBAS DE CAJA NEGRA – ANÁLISIS DE VALORES LÍMITES

VALORES LÍMITE (VÁLIDOS)		VALORES LÍMITE (INVÁLIDO)	
CD = TUC	CD = CBA	CD = DISTINTO DE TUC Y CBA	
CP = 1	CP = 60	CP = 0	CP = 61
ED = 1	ED = 99	ED = 0	ED = 100
ED = 65		ED = 66	ED = 64
CP = 5	CP = 7	CP = 4	CP = 8

VALORES LÍMITE (VÁLIDOS)		VALORES LÍMITE (INVÁLIDO)	
1 CD = TUC	2 CD = CBA	7 CD = DISTINTO DE TUC Y CBA	
3 CP = 1	4 CP = 60	8 CP = 0	9 CP = 61
5 ED = 1	6 ED = 99	10 ED = - 1	11 ED = 100
12 ED = 65		13 ED = 66	14 ED = 64
15 CP = 5	16 CP = 7	17 CP = 4	18 CP = 8

FUNDAMENTOS DE LA PROGRAMACIÓN

PRUEBAS DE CAJA NEGRA – ANÁLISIS DE VALORES LÍMITES

VALORES LÍMITE (VÁLIDOS)		VALORES LÍMITE (INVÁLIDO)	
1 CD = TUC	2 CD = CBA	7 CD = DISTINTO DE TUC Y CBA	
3 CP = 1	4 CP = 60	8 CP = 0	9 CP = 61
5 ED = 1	6 ED = 99	10 ED = 0	11 ED = 100
12 ED = 65		13 ED = 66	14 ED = 64
15 CP = 5	16 CP = 7	17 CP = 4	18 CP = 8

CD	CP	ED	CONDICIONES	RESULTADOS
TUC	1	1	1, 3, 5	Precio tarifa a Tucumán p/ 1 pasajero
CBA	60	99	2, 4, 6	Precio tarifa a Córdoba p/ 60 pasajeros
BS.AS.	1	1	7, 3 y 5	ERROR
TUC	0	99	1, 8 y 6	ERROR
CBA	61	1	2, 9 y 5	ERROR
CBA	1	0	2, 3 y 10	ERROR
TUC	60	100	1, 4 y 11	ERROR

FUNDAMENTOS DE LA PROGRAMACIÓN

PRUEBAS DE CAJA NEGRA – ANÁLISIS DE VALORES LÍMITES

12 ED = 65	13 ED = 66 14 ED = 64
15 CP = 5 16 CP = 7	17 CP = 4 18 CP = 8

CD	CP	ED	CONDICIONES	RESULTADOS
TUC	1	1	1, 3, 5	Precio tarifa a Tucumán p/ 1 pasajero
CBA	60	99	2, 4, 6	Precio tarifa a Córdoba p/ 60 pasajeros
BS.AS.	1	1	7, 3 y 5	ERROR
TUC	0	99	1, 8 y 6	ERROR
CBA	61	1	2, 9 y 5	ERROR
CBA	1	0	2, 3 y 10	ERROR
TUC	60	100	1, 4 y 11	ERROR
TUC	1	65	1, 4 y 12	Precio tarifa a Tucumán p/ 1 pasajero con 40% Desc.
TUC	1	66	1, 4 y 13	Precio tarifa a Tucumán p/ 1 pasajero
TUC	1	64	1, 4 y 14	Precio tarifa a Tucumán p/ 1 pasajero
CBA	5	99	2, 15 y 6	Precio tarifa a Córdoba p/ 5 pasajeros con 15 % Desc.
CBA	7	65	2, 16 y 12	Precio tarifa a Córdoba p/ 7 pasajeros con 15 % Desc. Y 40% Des por Eda
CBA	4	99	2, 17 y 6	Precio tarifa a Córdoba p/ 4 pasajeros
CBA	8	99	2, 18 y 6	Precio tarifa a Córdoba p/ 8 pasajeros

BIBLIOGRAFIA

- **Ingeniería de software**. Un enfoque práctico; Roger Pressman, sesta edición
Cap. 5 pag. 123 a 125.; Cap 13; Cap. 14

- **Ingeniería de software.** Un enfoque práctico; Roger Pressman, séptima edición
Cap. 4 pag. 94 a 96.; Cap 17; Cap. 18