**Universidad ORT Uruguay**

**Facultad de Ingeniería**

**Bernard Wand Polak**

**Diseño de Aplicaciones 1**

**Obligatorio 1**

**Nicolas Rodriguez**

**Nro. Est. 178370**

**Grupo N4A**

**Docente: Eduardo Cuñarro**

**Formulário de Antecedentes**

**Curso Diseño de Aplicaciones**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nro. Estudiante** | 178370 | |  | | --- | | foto del Estudiante | |
| **Nombre:** | Nicolas |
| **Apellido:** | Rodriguez |
| **Grupo / Turno:** | N4A |
| **¿Trabaja en algo relacionado con la carrera?**  **¿Qué tareas desempeña?** | SI.  Consultoría en Informática |
|  |  |

Tabla de contenido

[1. Descripción General 3](#_Toc496027723)

[1.1. Requerimiento 3](#_Toc496027724)

[2. Analisis de Requerimientos 3](#_Toc496027725)

[2.1. Requerimientos Funcionales 3](#_Toc496027726)

[2.1.1. Gestión de Alumnos 3](#_Toc496027727)

[2.1.2. Gestión de Materias 4](#_Toc496027728)

[2.1.3. Gestión de Docentes 5](#_Toc496027729)

[2.1.4. Gestión de Camionetas 6](#_Toc496027730)

[2.1.5. Gestión de Actividades 7](#_Toc496027731)

[2.2. Requerimientos No Funcionales 8](#_Toc496027732)

[2.2.1. Idioma 8](#_Toc496027733)

[2.2.1. Tecnologías y herramientas de desarrollo 8](#_Toc496027734)

[2.2.2. Mantenibilidad 8](#_Toc496027735)

[2.3. Casos de Uso 9](#_Toc496027736)

[3. Justificación de Diseño 20](#_Toc496027737)

[3.1. Mecanismos Generales 20](#_Toc496027738)

[3.2. Descripción de Decisiones 20](#_Toc496027739)

[4. Diagrama de Paquetes 20](#_Toc496027740)

[5. Diagrama de Clases 21](#_Toc496027741)

[5.1. Prueba 21](#_Toc496027742)

[5.2. Dominio 21](#_Toc496027743)

[5.3. Grafica 22](#_Toc496027744)

[6. Evidencia de Clean Code 22](#_Toc496027745)

[7. Datos de Prueba 22](#_Toc496027746)

[8. Resultados de Pruebas 23](#_Toc496027747)

# Descripción General

A continuación se muestra el documento donde se reflejan las distintas etapas, técnicas y recursos utilizados para la realización de la aplicación solicitada en el obligatorio 1 de Diseño de Aplicaciones 1.

Se realiza entonces la aplicación, desarrollada en lenguaje C# utilizando el entorno de desarrollo Visual Studio 2015. El versionado se realizó utilizando repositorios Git en la nube, lo que permite la ubicuidad y flexibilidad de dispositivos para trabajar, además del versionado y documentación. Se aplicaron las técnicas vistas en el curso TDD y Clean Code. La metodología TDD se aplicó de forma correcta siguiendo el ciclo, pero no para la totalidad de las clases del dominio debido a una restricción de tiempos físicos para poder cumplir con las fechas de entrega solicitadas. La metodología Clean Code se aplica en todo el desarrollo, ampliándose de forma más detallada la evidencia de su utilización en el punto 6 (Evidencia de Clean Code).

Es de destacar que la funcionalidad solicitada se cumple en su mayoría, quedando pendiente la completitud de la implementación del módulo de pagos.

# Analisis de Requerimientos

En esta sección se especifican los requerimientos funcionales y no funcionales del sistema a desarrollar.

El objetivo de este proyecto es crear un sistema que cumpla la función de ERP (Enterprise Resource Planner) que permita gestionar y manejar de forma más eficiente las operaciones de un colegio.

## Características de los usuarios (Actores)

Para esta instancia no se crean diferentes perfiles, asumiendo que quien use la aplicación será un Administrador designado por el colegio, que tendrá potestades para crear, modificar, eliminar, listar, etc. Materias, Docentes, Alumnos, Camionetas, Actividades y Pagos.

## Requerimientos Funcionales

### Gestión de Alumnos

**RF1 – Dar de Alta un Alumno**

Descripción: se ingresa un nombre, un apellido, un identificador numérico y un numero de cedula sin puntos ni guiones.

Actor: Administrador

Prioridad: Alta

**RF2 – Dar de Baja un Alumno**

Descripción: se ingresa el identificador numérico del alumno y se busca que exista en el sistema.

En caso de existir, se muestra el botón para confirmar el borrado. En caso contrario se despliega el mensaje de que no existe un alumno con ese identificador.

Actor: Administrador

Prioridad: Alta

**RF3 – Modificar un Alumno**

Descripción: se ingresa el identificador numérico del alumno y se busca que exista en el sistema.

En caso de existir, se muestran los campos necesarios con la información actual del usuario (Nombre, Apellido, Identificador, Numero de cedula) y el botón para confirmar la modificación. En caso contrario se despliega el mensaje de que no existe un alumno con ese identificador.

Actor: Administrador

Prioridad: Alta

**RF4 – Mostrar Listado de Alumnos**

Descripción: se despliega un panel sobre la izquierda con una lista con los alumnos existentes en el sistema. Al seleccionar un alumno haciendo clic sobre la línea de datos del alumno en el panel de la izquierda, se despliegan en el panel contiguo sobre la derecha el listado de materias a las que está inscripto.

Actor: Administrador

Prioridad: Alta

### Gestión de Materias

**RF5 – Dar de Alta una Materia**

Descripción: se ingresa un nombre y un código y se hace clic en el botón crear.

Actor: Administrador

Prioridad: Alta

**RF6 – Dar de Baja una Materia**

Descripción: se ingresa el código de la materia y se busca que exista en el sistema.

En caso de existir, se muestra el botón para confirmar el borrado. En caso contrario se despliega el mensaje de que no existe una materia con ese código.

Actor: Administrador

Prioridad: Alta

**RF7 – Modificar una Materia**

Descripción: se ingresa el código de la materia y se busca que exista en el sistema.

En caso de existir, se muestran los campos necesarios con la información actual de la materia (Nombre y código) y el botón para confirmar la modificación. En caso contrario se despliega el mensaje de que no existe una materia con ese código.

Actor: Administrador

Prioridad: Alta

**RF8 – Mostrar Listado de Materias**

Descripción: se despliega un panel sobre la izquierda con una lista con las materias existentes en el sistema. Al seleccionar una materia haciendo clic sobre la línea de datos de la materia en el panel de la izquierda, se despliegan en los paneles contiguos sobre la derecha el listado de alumnos inscriptos a dicha materia y el listado de docentes que dictan esa materia.

Actor: Administrador

Prioridad: Alta

### Gestión de Docentes

**RF9 – Dar de Alta un Docente**

Descripción: se ingresa un nombre y un apellido y se hace clic en el botón crear.

Actor: Administrador

Prioridad: Alta

**RF10 – Dar de Baja un Docente**

Descripción: se ingresa el apellido del docente y se busca que exista en el sistema.

En caso de existir, se muestra el botón para confirmar el borrado. En caso contrario se despliega el mensaje de que no existe un docente con ese apellido en el sistema.

Actor: Administrador

Prioridad: Alta

**RF11 – Modificar un Docente**

Descripción: se ingresa el apellido del docente y se busca que exista en el sistema.

En caso de existir, se muestran los campos Nombre y Apellido donde se realizan las modificaciones y además el botón para confirmar la modificación. En caso contrario se despliega el mensaje de que no existe un docente con ese apellido en el sistema.

Actor: Administrador

Prioridad: Alta

**RF12 – Mostrar Listado de Docentes**

Descripción: se despliega un panel sobre la izquierda con una lista con los docentes existentes en el sistema. Al seleccionar un docente haciendo clic sobre la línea de datos del docente en el panel de la izquierda, se despliegan en el panel contiguo sobre la derecha el listado de materias que dicta.

Actor: Administrador

Prioridad: Alta

### Gestión de Camionetas

**RF13 – Dar de Alta una Camioneta**

Descripción: se ingresa un nombre, una capacidad, numero identificador y se valida el checkbox en caso de estar disponible y se hace clic en el botón crear.

Actor: Administrador

Prioridad: Alta

**RF14 – Dar de Baja una Camioneta**

Descripción: se ingresa el número identificador de la camioneta y se busca que exista en el sistema.

En caso de existir, se muestra el botón para confirmar el borrado. En caso contrario se despliega el mensaje de que no existe una camioneta con ese número identificador en el sistema.

Actor: Administrador

Prioridad: Alta

**RF15 – Modificar una Camioneta**

Descripción: se ingresa el número identificador de la camioneta y se busca que exista en el sistema.

En caso de existir, se muestran los campos Nombre, Capacidad, Identificador y el checkbox de disponibilidad donde se realizan las modificaciones y además el botón para confirmar la modificación. En caso contrario se despliega el mensaje de que no existe una camioneta con ese número identificador en el sistema.

Actor: Administrador

Prioridad: Alta

**RF16 – Mostrar Listado de Camionetas Disponibles y Rutas**

Descripción: se despliega un panel sobre la izquierda con una lista con las camionetas disponibles existentes en el sistema. Al seleccionar una camioneta haciendo clic sobre la línea de datos de la misma en el panel de la izquierda, se despliegan en el panel contiguo sobre la derecha el listado de alumnos ordenados por distancia de menor a mayor que podrá recoger de acuerdo a su capacidad. Es decir, si la camioneta disponible tiene capacidad para 2 alumnos, y tomando como punto de partida de la camioneta siempre la posición de la escuela como 0,0, se despliegan los datos de los 2 alumnos que están más cercanos.

Actor: Administrador

Prioridad: Alta

### Gestión de Actividades

**RF17 – Dar de Alta una Actividad**

Descripción: se ingresa un nombre, un número identificador, se elige una fecha y se ingresa un monto entero sin puntos ni comas y se hace clic en el botón crear.

Actor: Administrador

Prioridad: Alta

**RF18 – Dar de Baja una Actividad**

Descripción: se ingresa el número identificador de la actividad y se busca que exista en el sistema.

En caso de existir, se muestra el botón para confirmar el borrado. En caso contrario se despliega el mensaje de que no existe una actividad con ese número identificador en el sistema.

Actor: Administrador

Prioridad: Alta

**RF19 – Modificar una Actividad**

Descripción: se ingresa el número identificador de la actividad y se busca que exista en el sistema.

En caso de existir, se muestran los campos Nombre, Identificador, Fecha y costo y además el botón para confirmar la modificación. En caso contrario se despliega el mensaje de que no existe una actividad con ese número identificador en el sistema.

Actor: Administrador

Prioridad: Alta

**RF20 – Mostrar Listado de Alumnos por Actividad**

Descripción: se despliega un panel sobre la izquierda con una lista con las actividades existentes en el sistema. Al seleccionar una actividad haciendo clic sobre la línea de datos de la misma en el panel de la izquierda, se despliegan en el panel contiguo sobre la derecha el listado de alumnos que participaran en la actividad seleccionada.

Actor: Administrador

Prioridad: Alta

A priori la funcionalidad requerida no se pudo testear dado que la aplicación no corría. Revisando el contenido del repositorio se detectó que lo que se entregó no fue en formato release. De todas maneras se quiso probar en debug, pero al momento de ejecutar el proyecto en visual se observo que faltaban 3 clases (DeviceTypeValidator.cs, DeviceValidator.cs, VariableValidator.cs) Se reviso el repositorio web para corroborar que no hubiera errores en la descarga y se observó que las clases faltantes tampoco figuraban allí. Se habló con un integrante del grupo para comentarle esta dificultad y quedamos en que enviaría las clases faltantes para probar la funcionalidad.

Otra observación fue que la branch “master” del repositorio, no era la más actual y la que debería tener todo el contenido solicitado en el obligatorio. El repositorio cuenta con 3 “branches” (master, Development, Alarm Feature). La branch “Alarm Feature” fue la que se consideró para la corrección, dado que es la más actual en el tiempo y a la que se le agregaron las clases que faltaban.

## Requerimientos No Funcionales

### Idioma

La codificación fue realizada en idioma ingles

### Tecnologías y herramientas de desarrollo

* Microsoft Visual Studio 2015 .NET (Lenguaje C#)
* Astah para modelado UML

### Mantenibilidad

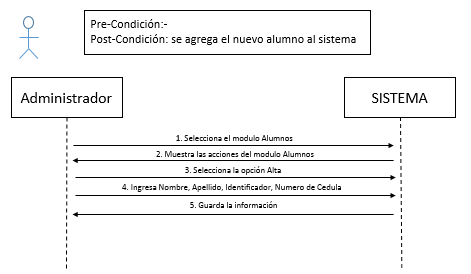
Utilización de repositorio Git para control de versionado y seguimiento

Desarrollo guiado por pruebas (TDD), realizando primero las pruebas y luego el refactoring correspondiente.

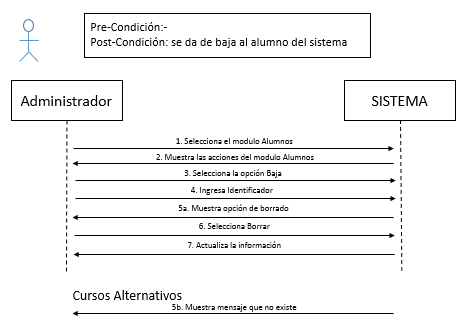
Se realizan todos los pasos correspondientes del ciclo Test-First Development y Refactoring (Red, Green, Refactor, Red, Green…)

## Casos de Uso

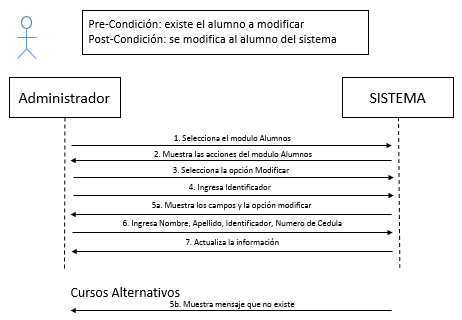
* Alta Alumno



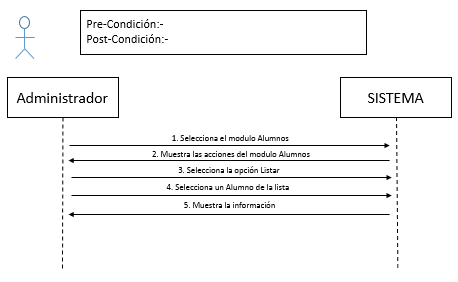
* Baja Alumno



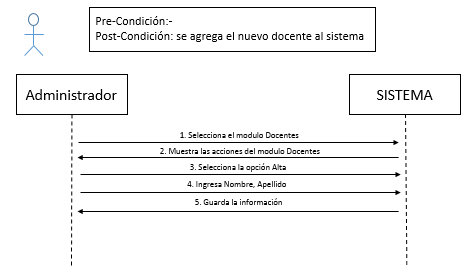
* Modificar Alumno



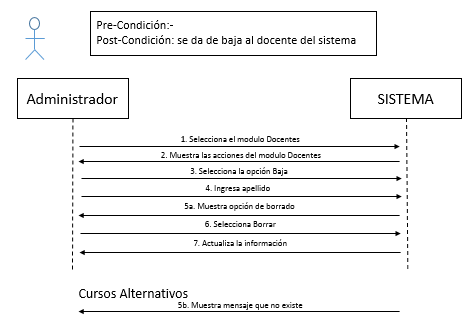
* Listar Alumnos y sus cursos



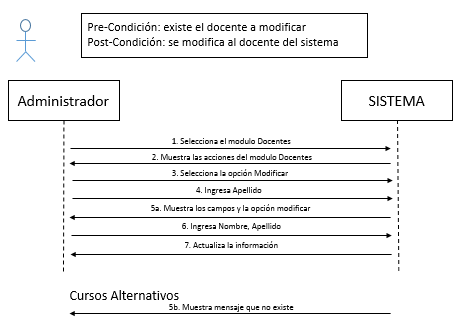
* Alta Docente



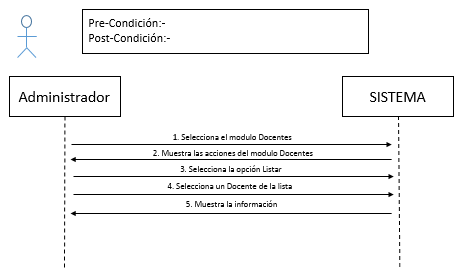
* Baja Docente



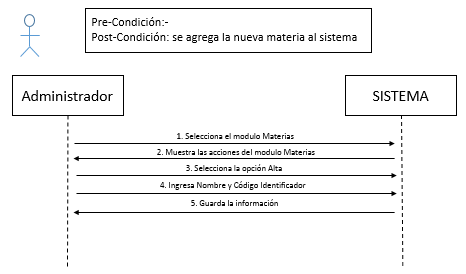
* Modificar Docente



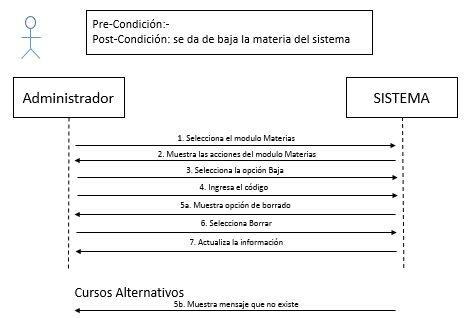
* Listar Docente



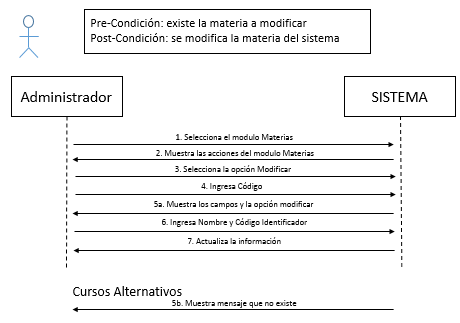
* Alta Materia



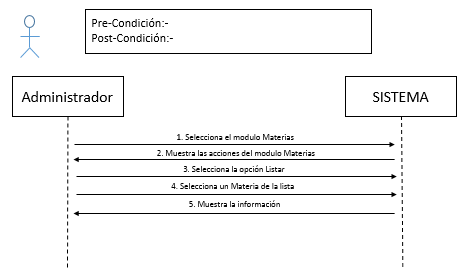
* Baja Materia



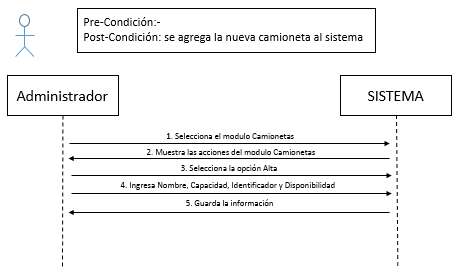
* Modificar Materia



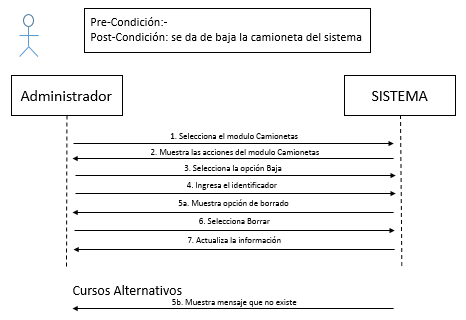
* Listar Materia



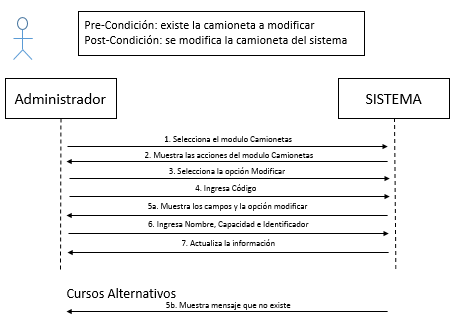
* Alta Camioneta



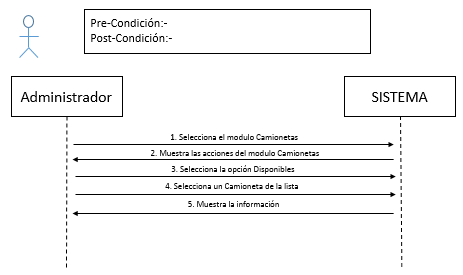
* Baja Camioneta



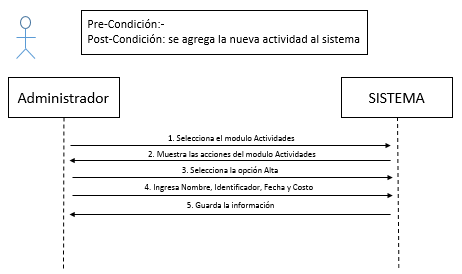
* Modificar Camioneta



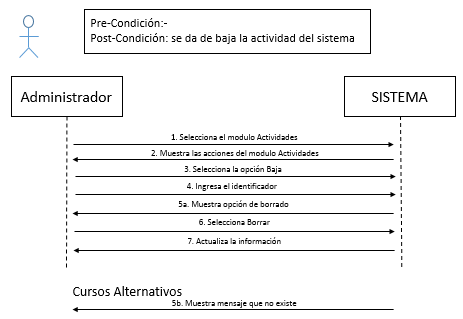
* Listar Camionetas Disponibles y Rutas



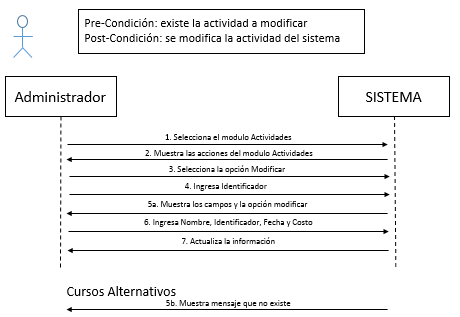
* Alta Actividad



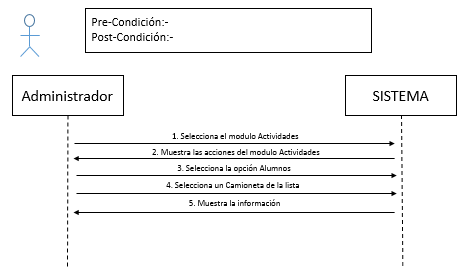
* Baja Actividad



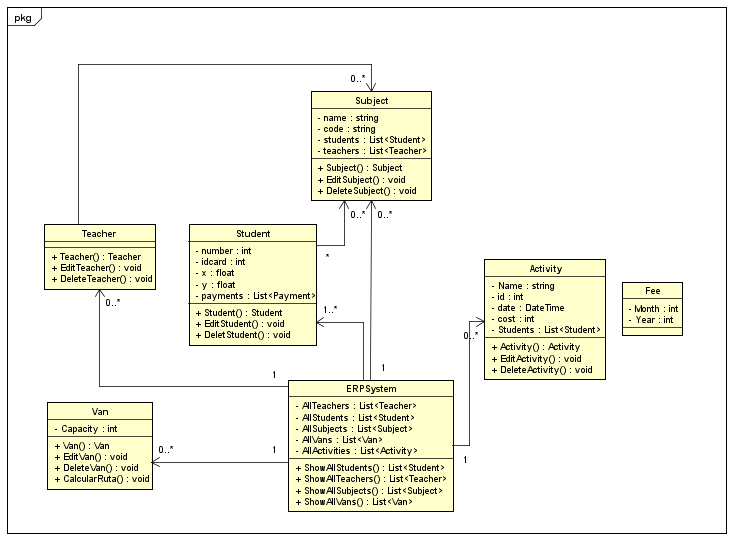
* Modificar Actividad



* Listar Alumnos participantes de cada Actividad



## Modelo Conceptual



# Justificación de Diseño

## Descripción de Decisiones

Se decidió realizar una clase abstracta Person dado que en el mapa conceptual se observaba que las clases Student y Teacher compartían atributos y comportamiento en común.

Se realizó también una clase abstracta Payment dado que se observó también que las clases Activity y Fee compartían también atributos y comportamiento.

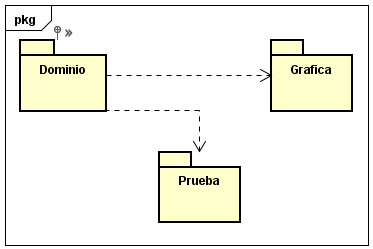
Se decidió calcular las distancias como la suma de la distancia entre la posición X del alumno y 0 (posición X de la escuela) y la distancia entre la posición Y del alumno y 0 (posición Y de la escuela), considerando que las camionetas se mueve en líneas rectas (sin diagonales) idealizando el “mapa” como un damero. No se consideran posiciones negativas, es decir, se asume que todas las posiciones x e y de cada alumno son mayores o iguales a 0.

Una “ruta” para una camioneta se representa como una lista de alumnos a recoger por dicha camioneta ordenada por la distancia (menor a mayor), calculada a partir de las coordenadas de cada alumno desde la escuela, para que dicha camioneta recorra la menor distancia posible.

Las “rutas” se calculan para las camionetas considerando primero las que estén disponibles y tengan mayor capacidad hasta que esta quede llena.

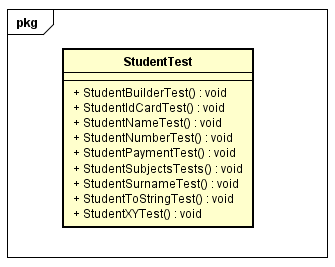
El módulo de Pagos está inconcluso. Esto no afecta el funcionamiento del resto de los módulos. Se planifica colocarlo dentro del módulo “Alumno”.

# Diagrama de Paquetes

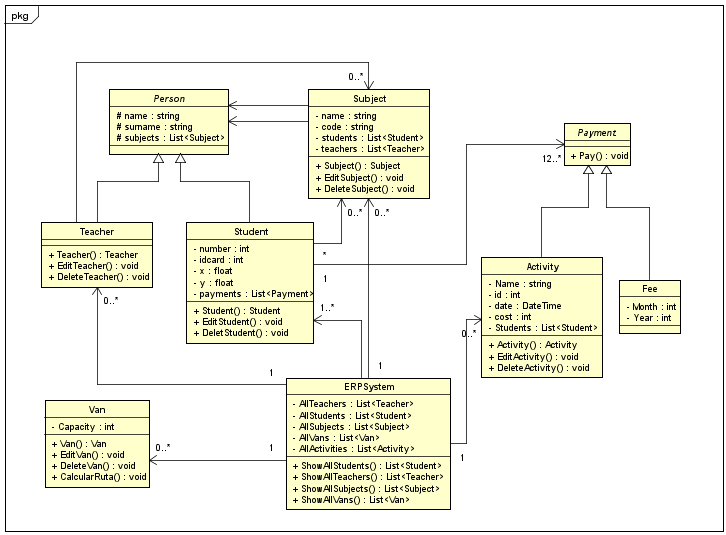


# Diagrama de Clases

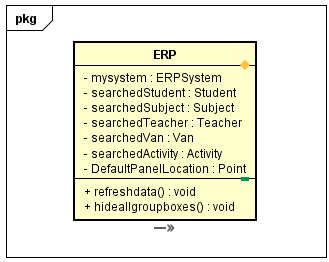
## Prueba



## Dominio



## Grafica



# Evidencia de Clean Code

Se aplica clean code nombrando a la mayoría de los elementos (clases, atributos, métodos, variables, etc.) de manera suficientemente significativa y descriptiva (ej. los botones del proyecto de interfaz gráfica tienen por nombre btn+Módulo+Acción → “btnTeacherDelete”). Adicionalmente no se tienen implementados métodos con más de 2 parámetros, lo que indica que los métodos hacen 1 sola cosa y bien, sin tener complejidades. Tampoco hay métodos con demasiada cantidad de líneas (más de 7), salvo el método de cargar datos de prueba en la clase ERPSystem, ni con demasiados niveles de indentación (no más de 2). Al aplicar esta metodología clean code, permite que el código sea más legible y entendible por el propio desarrollador y terceros, posibilitando hacer modificaciones de manera más ágil en caso de tener que incorporar correcciones o nuevas funcionalidades

# Datos de Prueba

Alumnos:

Se crean 4 alumnos con el mismo string de nombre que apellido (S1, S2, S3 y S4). Se identifican con 1 digito para la cedula de identidad numerados del 1 al 4. Análogamente para el identificador de número de estudiante.

Docentes:

Se crean 4 docentes con el mismo string de nombre que apellido (T1, T2, T3 y T4).

Materias:

Se crean 3 materias con el string de nombre “subject1”, “subject2” y “subject3”. Análogamente los códigos alfanuméricos de estas materias son “sub1”, “sub2” y “sub3”. Se agregan análogamente los profesores T1 y los alumnos S1 a las materias Subject1. Se repite el procedimiento para las restantes materias subject2 y subject3.

Actividades:

Se crean 3 actividades nombradas a1, a2 y a3, análogamente numeradas con el código 1, 2,y 3 y utilizando este mismo valor para su costo. La fecha definida para todas las actividades es la fecha actual del sistema al momento de darse de alta. A la actividad a1 se le agrega como participante al alumno S1 y análogamente se realiza lo mismo para las actividades a2 y a3.

Camionetas:

Se crean 3 camionetas nombradas v1, v2 y v3. Las capacidades son análogas a los id de cada una siendo estos análogos a los nombres (1,2 y 3). El valor booleano de disponibilidad es true para las 3 camionetas.

# Resultados de Pruebas Unitarias

La metodología TDD se utiliza para la clase Student del Dominio.

No se aplica a todas las clases del dominio por motivos de tiempos físicos disponibles al ser 1 solo integrante en el equipo.

La metodología se aplica y se entienden los beneficios que brinda. Al utilizar este método la cobertura de las pruebas es cercana a 100% dado que primero se crea la prueba, para luego hacer lo mínimo necesario para que compile, y posteriormente se realiza el refactoring para que la prueba pase. Esto asegura generar un código mínimo, prolijo y ordenado, que cumpla con el objetivo deseado. Este método también sirve para realizar un tracking del desarrollo, dado que cada paso que se realiza queda documentado, al realizar los commits correspondientes en el repositorio remoto Git. Esto ayuda en el troubleshooting en caso de ocurrencia de errores.

Se utilizaron pruebas automatizadas de Visual Studio Unit Tests. Las mismas se pueden evidenciar en el código dentro del paquete Prueba.

Análisis de cobertura

Se realiza el análisis de cobertura para el código implementado con TDD y el porcentaje de cobertura es de un 100% para la clase Student que es la que nos interesa en este caso particular. Esto condice con la teoría del método TDD. El análisis de cobertura se puede ver en el archivo que se adjunta “Cobertura.coveragexml”. También figuran en el repositorio los distintos analisis de coberturas realizados para modificaciones introducidas durante el desarrollo.

