

**Metodologías de programación II**

Smalltalk

**Alumnos: Acuña Nicolás, D’Amico Claudio, Ferreyra Enzo**

**Profesora: Cappelletti Claudia**

**Comisión: 1**

**TP Integrador: Smalltalk**

*INDICE*

*Descripción del proyecto*

............................................................................................ 2

*Diagrama UML*

*............................................................................................ 3*

*Diagrama de secuencias*

*............................................................................................ 4*

*Implementación y especificación de clases utilizadas*

*............................................................................................ 5*

*Implementación de la aplicación en Smalltalk*

............................................................................................ 6

*Metodología utilizada en el proyecto ............................................................................................ 7*

*Frameworks para desarrollar el proyecto en POO*

*........................................................................................... 8*

*Casos de usos*

*............................................................................................ 9*

*Patrones en el proyecto*

*........................................................................................... 10*

*Repositorios utilizados para el proyecto*

*………………………………………………………………………………..........11*

*Descripción del proyecto.*

*Para realizar este TP final se propuso la implementación en Smalltalk de un programa que simule el funcionamiento de un negocio implementando las funcionalidades vistas en clase del lenguaje de programación antes mencionado.*

*Esto implica implementar todos los recursos vistos y especial utilizar algunos cuando sea más oportuno para los integrantes del grupo, algunos de los que se pide en más detalle son: select, collect, reject, detect y además un diccionario que cuente las ocurrencias de una determinada situación, además se debe mostrar al menos una vez datos de forma ordenada.*

*Para realizar este proyecto se pensó en realizar un programa para gestionar un negocio de remiseria, dado que analizando entre los integrantes del grupo que desarrollara el sistema, creemos que tiene todo lo necesario para poder implementar todos los recursos de Smalltalk como se pidió en el trabajo integrador.*

*Este sistema contara con la funcionalidad de dar de alta tanto a clientes, choferes o recepcionistas de la remiseria, así como también poder modificarlos, buscarlos o eliminarlos.*

*También contara con la funcionalidad de poder guardar los datos de los viajes como también de las reservas que se realizaran, estas también se podrán buscar, modificar y eliminar según la necesidad del negocio.*

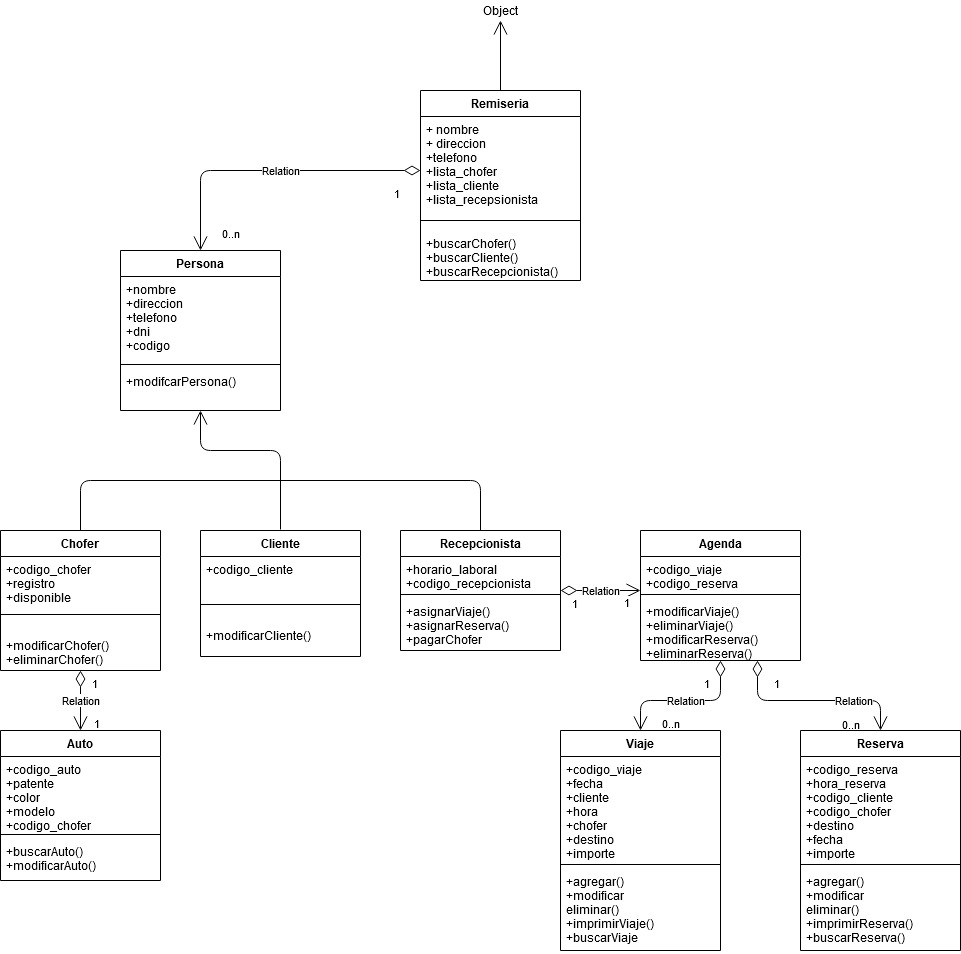
*Todos los datos de viajes y reservas se guardarán en una agenda que manejara la recepcionista para así poder ser accedida y poder realizar las consultas necesarias.*

*Por ultimo cada chofer tendrá su propio automóvil con sus respectivos datos los cuales también podrán ser buscados, modificados o eliminados según la necesidad del negocio.*

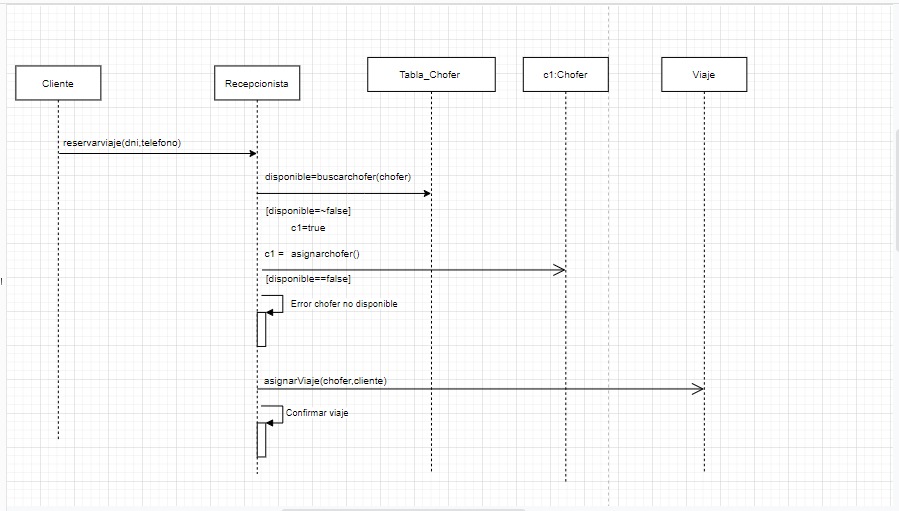
*Esta y todas las clases y métodos implementados serán detallados más adelante cada una con su respectiva imagen, así se podrá tener una mejor visión del programa realizado.*

*El sistema que se realizara constara de 4 clases funcionales descriptas en el diagrama UML que se presentara a continuación, se elegirán las más importantes para poder demostrar que el sistema funciona y muestra lo pedido en el TP integrador, por este motivo a continuación presentaremos cada una de las partes vitales de este sistema.*

*Diagrama UML.*

**

*Diagrama de secuencias.*



*Clases y métodos del proyecto.*

*Metodología Scrum en el proyecto.*

*Para realizar este proyecto se eligió la metodología de trabajo Scrum que ahora pasaremos a detallar:*

*Nos basaremos en tres pilares:*

*1. Transparencia*

*Con el método Scrum todos los implicados tienen conocimiento de qué ocurre en el proyecto y cómo ocurre. Esto hace que haya un entendimiento “común” del proyecto, una visión global.*

*2. Inspección*

*Los miembros del equipo Scrum frecuentemente inspeccionan el progreso para detectar posibles problemas. La inspección no es un examen diario, sino una forma de saber que el trabajo fluye y que el equipo funciona de manera auto-organizada.*

*3. Adaptación*

*Cuando hay algo que cambiar, el equipo se ajusta para conseguir el objetivo del sprint. Esta es la clave para conseguir el éxito en proyectos complejos, donde los requisitos son cambiantes o poco definidos y en donde la adaptación, la innovación, la complejidad y flexibilidad son fundamentales.*

*Nos basaremos en tres roles fundamentales:* ***Product Owner (cabe aclarar que en este caso será la profesora de la materia), Scrum Master y Equipo de desarrollo.***

***Vamos a realizar el proyecto dividiéndolo en tres tramos a los cuales llamaremos Sprints, estos contendrán desarrollos iterativos que son: sprint planning, daily meeting, sprint review y sprint retrospective.***

***Como hemos mencionado se desarrollará nuestro proyecto en tres sprint, cada uno durará como máximo una semana.***

*Sprint.*

***En cada Sprint se procederá a tener una iteración para obtener la información necesaria a fin de desarrollar la aplicación.***

***El primer sprint dará comienzo el 19/10/2020 y durará hasta el 26/10/2020, en este Sprint se abordará: Comienzo de la aplicación, comienzo de diagramas UML de clases, de secuencias y documentación pertinente a cada uno de ellos, al final el entregable deberá se deberá contar con toda la documentación para el comienzo del código del proyecto y sus clases creadas.***

***El segundo Sprint dará comienzo el 26/10/2020 y finalizará el 02/11/2020, en este Sprint se abordará: Implementación del repositorio del proyecto, clases e instancias terminadas, documentación terminada y comienzo del informe.***

***El tercer Sprint dará comienzo el 02/11/2020 y finalizará el 09/11/2020, en este último Sprint se abordará: Finalización y funcionalidad total de la aplicación, finalización del informe y presentación del proyecto.***

*Sprint Planning.*

*Para cada comienzo de Sprint se realizará un Sprint Planning, en esta reunión todo el equipo Scrum define qué tareas se van a abordar y cuál será el objetivo del* Sprint*. La primera reunión que se hace en el* sprint ***puede llegar a tener una duración de 2 horas para Sprint de una semana.***

***Daily Meeting.***

***Luego se tendrán las Daily meeting,*** *esta es una reunión diaria dentro del sprint que tiene* ***como máximo 15 minutos de duración****. En ella debe participar, sí o sí, el* ***equipo de desarrollo y el Scrum Master.*** *El Product Owner no tienen necesidad de estar presente.*

*Sprint Rewiew.*

*Tendremos también los Sprint rewiew, aquí la review del valor que vamos a entregar al cliente se hace en esta reunión, al final de cada sprint. Su duración es de* ***2 horas para sprints de una semana,*** *y es la única reunión de Scrum a la que puede asistir el cliente. En ella el Product Owner* ***presenta lo desarrollado al cliente*** *y el equipo de desarrollo* ***muestra su funcionamiento.*** *El cliente valida los cambios realizados y además brinda feedback sobre nuevas tareas que el Product Owner tendrá que agregar al Product backlog.*

*Sprint Retrospective.*

*La retrospectiva es el último evento de Scrum y tiene una duración de 2 horas para Sprints de una semana, y es la reunión del equipo en la que se hace una evaluación de cómo se ha implementado la* ***metodología Scrum*** *en el último* sprint*.*

*Es una gran oportunidad para el* ***equipo Scrum*** *de inspeccionarse a sí mismo, proponiendo mejoras para el siguiente sprint.*

*El resultado: una lista de mejoras que debe aplicar el siguiente día, ya que, al finalizar la retrospectiva, inmediatamente comienza un nuevo* sprint*, que incluye el* sprint planning, daily meeting, sprint review *y el ya mencionado sprint* retrospective.

***El primer Sprint del 19/10/2020 hasta el 26/10/2020.***

***Sprint Backlog*.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Identificador (ID) de la Historia | Enunciado de la Historia | Estado | Iteración (Sprint) | Prioridad |
| 1 | Diagrama UML | Iniciada | 1 | Alta |
| 2 | Diagrama de Secuencias | Iniciada | 1 | Alta |
| 3 | Documentación | Iniciada | 1 | Alta |
| 4 | Clase Remiseria | Iniciada | 1 | Alta |
| 5 | Clase Chofer | Iniciada | 1 | Alta |
| 6 | Clase Cliente | Iniciada | 1 | Alta |
| 7 | Clase Recepcionista | Iniciada | 1 | Alta |
| 8 | Clase Agenda | Iniciada | 1 | Alta |

***El segundo Sprint del 26/10/2020 hasta el 02/11/2020.***

***Sprint Backlog***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Identificador (ID) de la Historia | Enunciado de la Historia | Estado | Iteración (Sprint) | Prioridad |
| 9 | Clase viajes | Iniciada | 1 | Alta |
| 10 | Repositorio remoto | Iniciada | 1 | Alta |
| 11 | Fin Documentación | Iniciada | 1 | Alta |
| 12 | crearPersonaNom | Iniciada | 1 | Alta |
| 13 | crearCoferNom | Iniciada | 1 | Alta |
| 14 | crearClienteNom | Iniciada | 1 | Alta |
| 15 | crearRecepcionistaNom | Iniciada | 1 | Alta |
| 16 | Comienzo del Informe | Iniciada | 1 | Alta |

***El tercer Sprint del 02/11/2020 hasta el 09/11/2020.***

***Sprint Backlog***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Identificador (ID) de la Historia | Enunciado de la Historia | Estado | Iteración (Sprint) | Prioridad |
| 17 | Finalización de Clases | Iniciada | 1 | Alta |
| 18 | Finalización de métodos | Iniciada | 1 | Alta |
| 19 | Pruebas del sistema | Iniciada | 1 | Alta |
| 20 | Pruebas del repositorio | Iniciada | 1 | Alta |
| 21 | Análisis funcional | Iniciada | 1 | Alta |
| 22 | Finalización del Sistema | Iniciada | 1 | Alta |
| 23 | Finalización del informe | Iniciada | 1 | Alta |
| 24 | Presentación del Proyecto | Iniciada | 1 | Alta |

*Product Backlog.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Identificador (ID) de la Historia | Enunciado de la Historia | Estado | Iteración (Sprint) | Prioridad |
| 1 | Diagrama UML | Iniciada | 1 | Alta |
| 2 | Diagrama de Secuencias | Iniciada | 1 | Alta |
| 3 | Documentación | Iniciada | 1 | Alta |
| 4 | Clase Remiseria | Iniciada | 1 | Alta |
| 5 | Clase Chofer | Iniciada | 1 | Alta |
| 6 | Clase Cliente | Iniciada | 1 | Alta |
| 7 | Clase Recepcionista | Iniciada | 1 | Alta |
| 8 | Clase Agenda | Iniciada | 1 | Alta |
| 9 | Clase Viajes | Iniciada | 2 | Alta |
| 10 | Repositorio remoto | Iniciada | 2 | Alta |
| 11 | Fin Documentación | Iniciada | 2 | Alta |
| 12 | crearPersonaNom | Iniciada | 2 | Alta |
| 13 | crearChoferNom | Iniciada | 2 | Alta |
| 14 | crearClienteNom | Iniciada | 2 | Alta |
| 15 | crearRecepsionistaNom | Iniciada | 2 | Alta |
| 16 | Comienzo del Informe | Iniciada | 3 | Alta |
| 17 | Finalización de Clases | No inicio | 3 | Alta |
| 18 | Finalización de métodos | No inicio | 3 | Alta |
| 19 | Pruebas del sistema | No inicio | 3 | Alta |
| 20 | Pruebas del repositorio | No inicio | 3 | Alta |
| 21 | Análisis funcional | No inicio | 3 | Alta |
| 22 | Finalización del sistema | No inicio | 3 | Alta |
| 23 | Finalización del Informe | No inicio | 3 | Alta |
| 24 | Presentación del Proyecto | No inicio | 3 | Alta |

*Frameworks para desarrollar el proyecto en POO.*

*En esta sección pasaremos a describir algunos Frameworks que se podrían utilizar para realizar un desarrollo orientado a objetos.*

*.Net: Esta plataforma fue construida aplicando el paradigma de Programación Orientada a Objetos (POO). El núcleo de lenguajes como C# están basados en los principios OO.*

*Ruby: Es un lenguaje de programación de propósito* ***general****, es decir, con Ruby se pueden desarrollar todo tipo de aplicaciones diferentes: aplicaciones de servicio web, clientes de correo electrónico, procesamiento de datos en Backend, aplicaciones de red, etc. Es 100% orientado a objetos.*

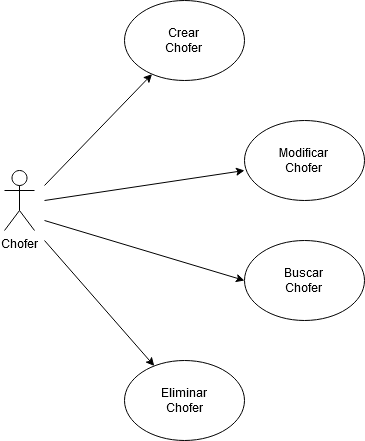
*Zend: Este es un framework de PHP basado en MVC,(modelo vista controlador)* [*orientado a objetos*](https://es.wikipedia.org/wiki/Programaci%C3%B3n_orientada_a_objetos)*, que permite cargar solo los componentes que se requieren como bibliotecas individuales.*

*Flutter: Es un Framework que utiliza Dart que es un lenguaje de programación orientado a objetos y creado por Google. Es altamente versátil, por lo cual puede ser utilizado en el desarrollo de aplicaciones mobile y desktop, scripts, backend y más.*

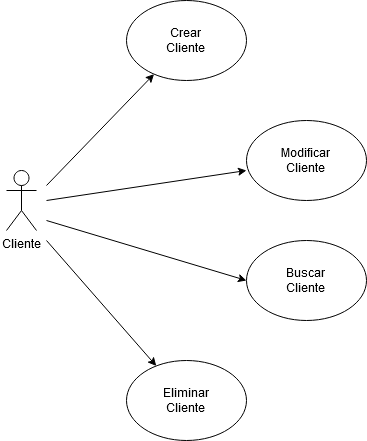
*Spring: Se trata de un* framework*de desarrollo de aplicaciones ligero para Java EE que en realidad se puede utilizar para todo tipo de aplicaciones, no sólo para la Web, pero que es especialmente útil para éstas utilizando Poo.*

*Casos de usos*

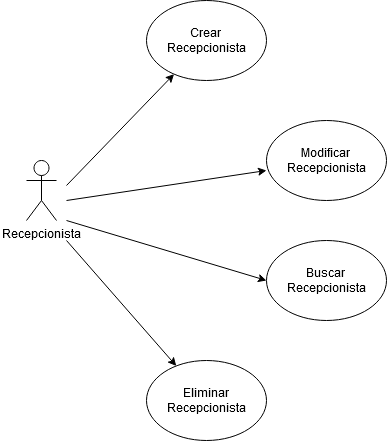
*Chofer*

**

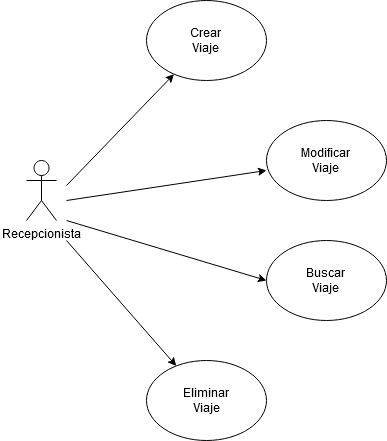
*Cliente*

**

*Recepcionista*

**

*Viaje*

**

*Patrones en el proyecto*

*Los patrones de diseño que se pueden utilizar en el sistema son varios, pero el equipo de desarrollo eligió dos que serían de gran utilidad a la hora de implementar el sistema.*

*Factory Method: Este patrón consiste en tener una clase a la cual delegar la responsabilidad de la creación de los objetos, es decir este patrón de Diseño tiene como función abordar el problema de crear objetos sin tener que especificar la clase exacta a la que han de pertenecer y sin tener que acceder directamente a la lógica de creación.*

*De esta manera utilizaríamos este patrón para la creación de los objetos Chofer, Cliente y Recepcionista la cual también tendrá sus propias características aparte de la que hereda a través de la interfaz Factory.*

*Singleton: El patrón de diseño Singleton (instancia única) está diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. Este patrón es aplicable en sistemas en los que se desea poder garantizar que solo existe una instancia de una clase.*

*Siempre que se crea un objeto nuevo (en Java, con la palabra reservada new) se invoca al constructor del objeto para que cree una instancia. Por lo general, los constructores son públicos. El Singleton lo que hace es convertir el constructor en privado, de manera que nadie lo pueda instanciar.*

*De esta manera con este patrón garantizamos que el objeto Remiseria sea instanciado una sola vez, quedando este dentro de un método privado al cual no se tiene acceso desde otro objeto.*

*Estos son los dos patrones que se implementarían en el sistema y creemos que son los que nos simplificarían a la hora de la implementación.*

*Repositorio del Proyecto.*

*Para realizar el proyecto se utilizó como repositorio GitHub ya que el equipo de desarrollo contaba con experiencia utilizándolo.*

*La dirección del proyecto es: https://github.com/claudiodamico/TP-Metodologias-II*