Universidad Nacional de Costa Rica

Facultad de Ciencias Exactas y Naturales

Escuela de Informática

Bidkar Solís Pérez

Bryan Murillo

Proyecto 3:

Vehicle Authentication Authority

Arquitectura de información

Año 2018

II Ciclo

# Introducción

Con el pasar de los años las prácticas comunes de desarrollo de software llevan al programador a escribir código tratando de comprender primero lo que los clientes desean. Este tipo de desarrollo resulta en procesos caros y demoras de entrega de los proyectos. Conforme se va aprendiendo de los errores de las malas prácticas de desarrollo en el mundo del software cada vez un diseño del sistema se vuelve más importante junto con su maquetación.

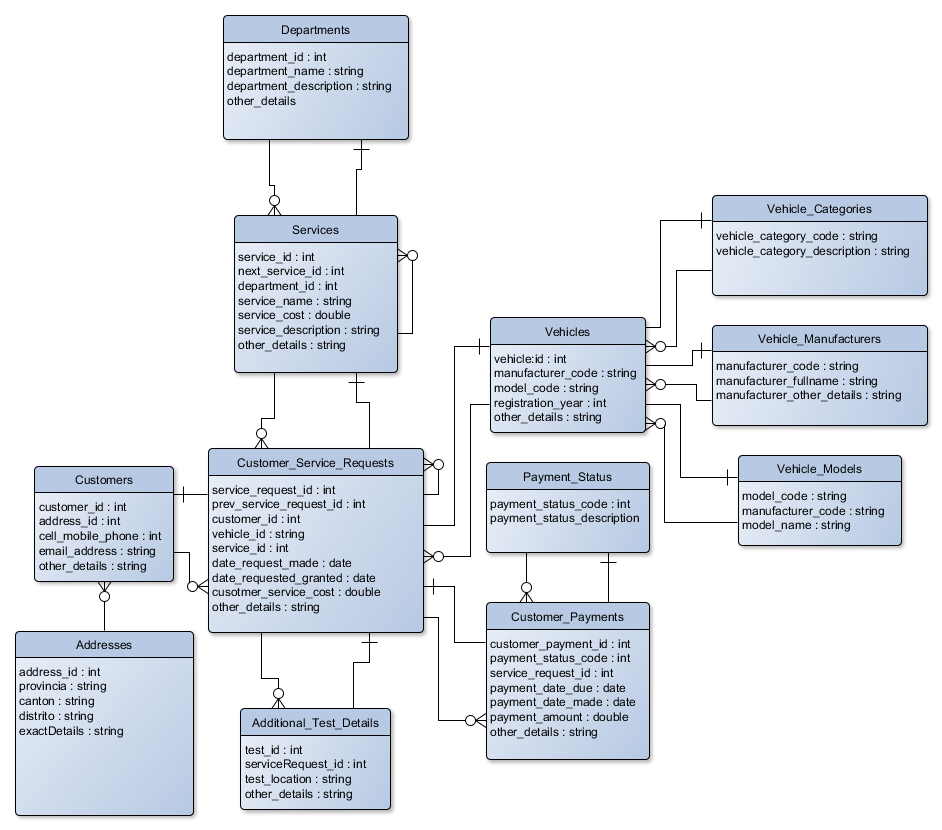
Así de una manera visual el producto es aprobado por el cliente antes de empezar a realizar código que más adelante en un proyecto puede ser desechado. Y puede que este código tomo horas de horas por parte del grupo de desarrollo y al final no sirve de nada. O incluso por puede resultar en un código de alto grado de aceptación y eficiencia pero la parte visual no es del agrado del cliente.

Es por ello que el análisis de diseño de software se vuelve demasiado importante en el proceso de desarrollo. Esto realizando el proceso de maquetación y así la parte visual es aprobada por los clientes mucho antes de iniciar el desarrollo, y aún si este diseño requiere de alguna modificación el impacto es menor ya que se tiene una base aprobada.

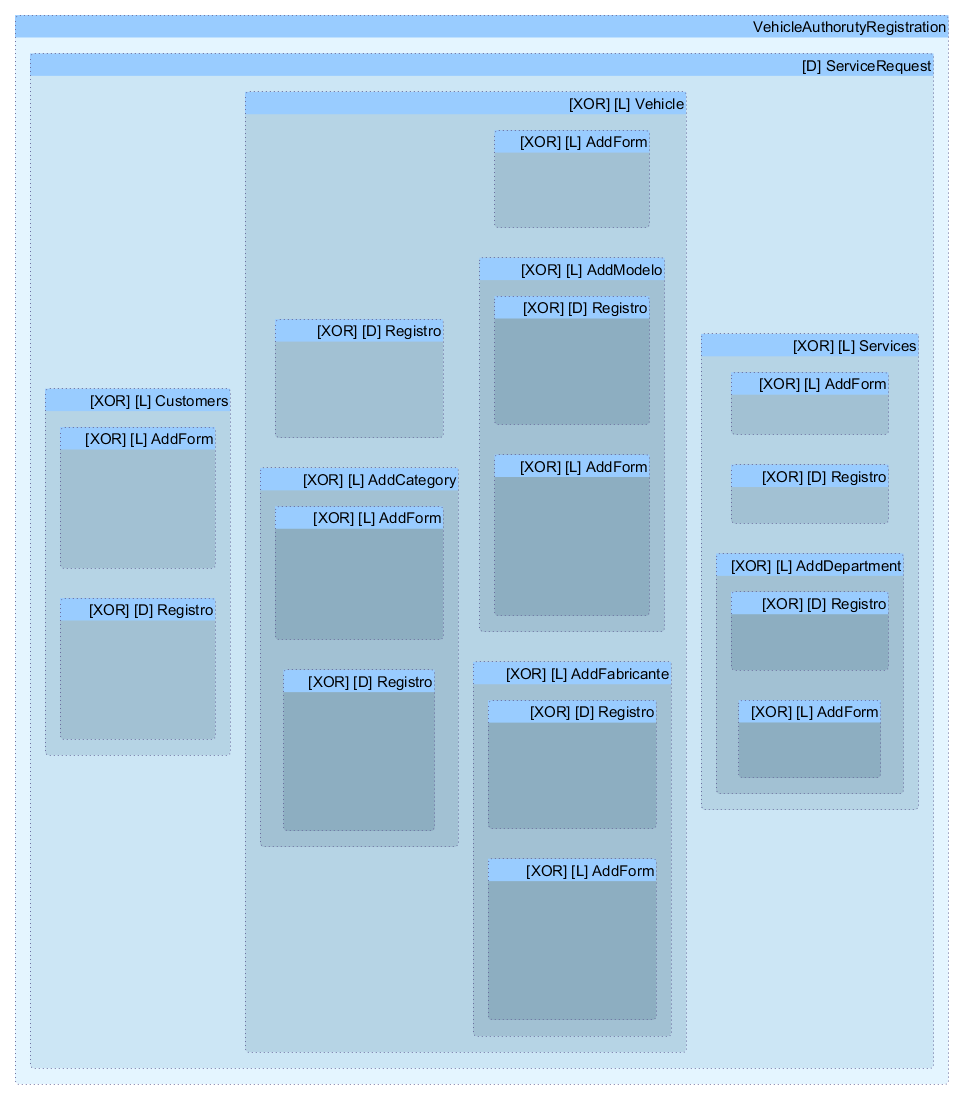
Pero la parte visual es para los clientes, ahora ¿Cómo desarrollar de una manera visual y lógica el control de los contenedores y sus elementos en conjunto con sus acciones y respuestas? Es ahí donde entran los sistemas de modelación de software en el cual se traducen lo visual y se agrega la parte lógica para ser desarrollada en código funcional.

A continuación se presenta un diseño básico bajo el modelo de *Interactive Flow Modeling Lanaguage* aplicado a una aplicación a un modelo de datos dado acerca de un monitoreo se servicios a vehículos. Los modelos a presentar son: modelo de dominio, el modelo de composición y modelo de acciones.

Modelo de datos

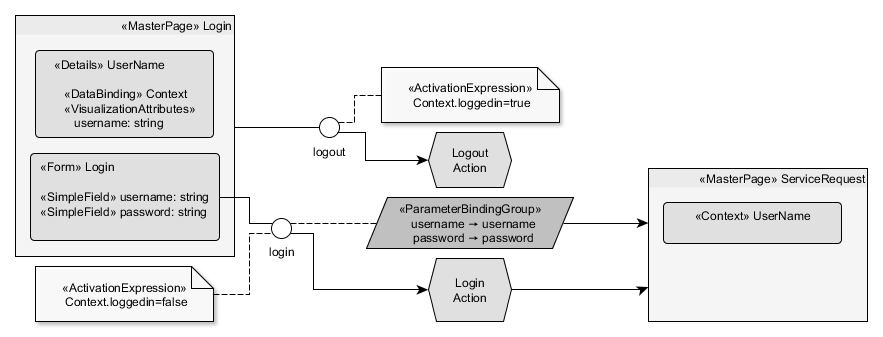


Modelo composición:

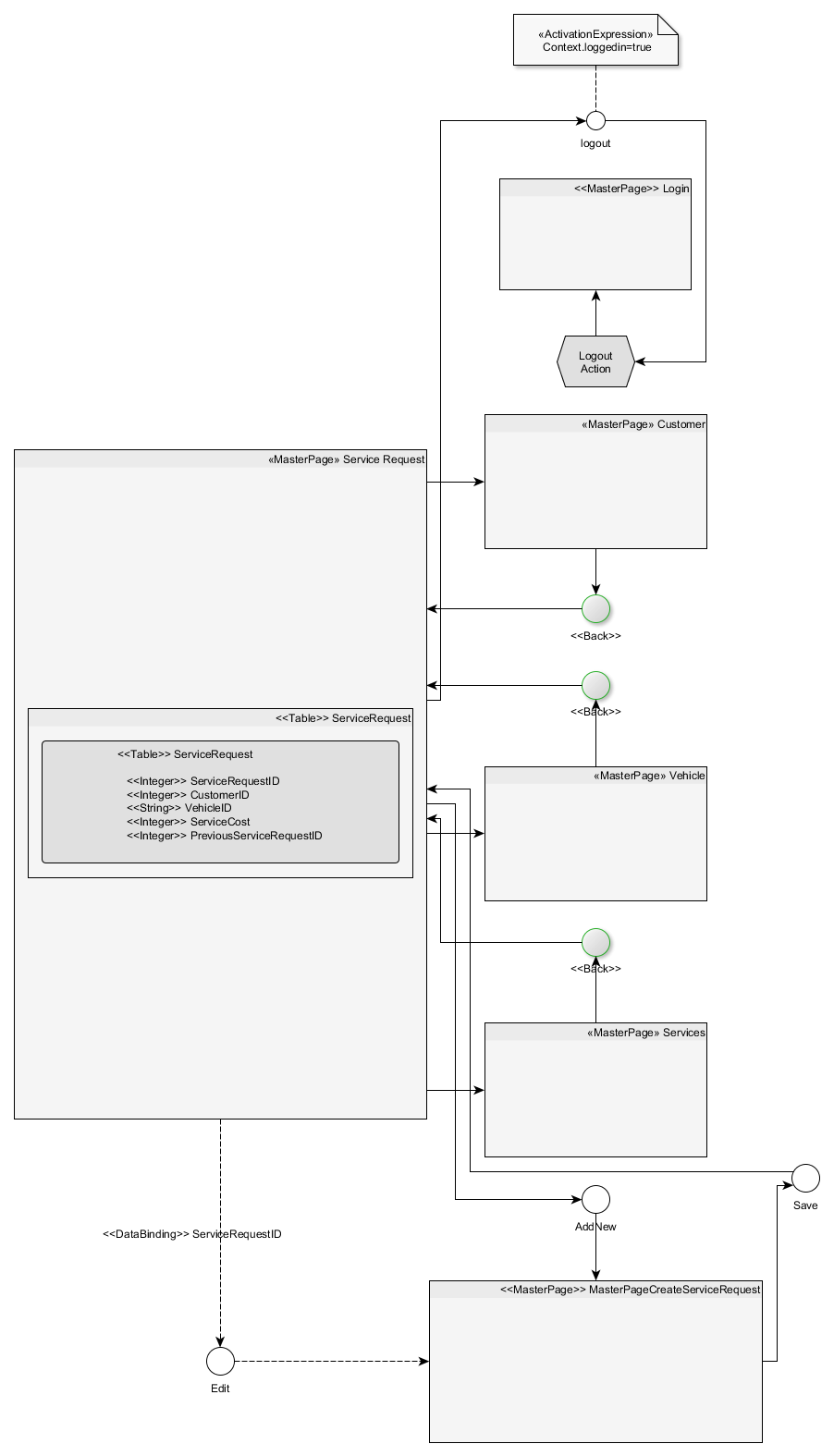


Modelos de contenido/navegación/acciones:

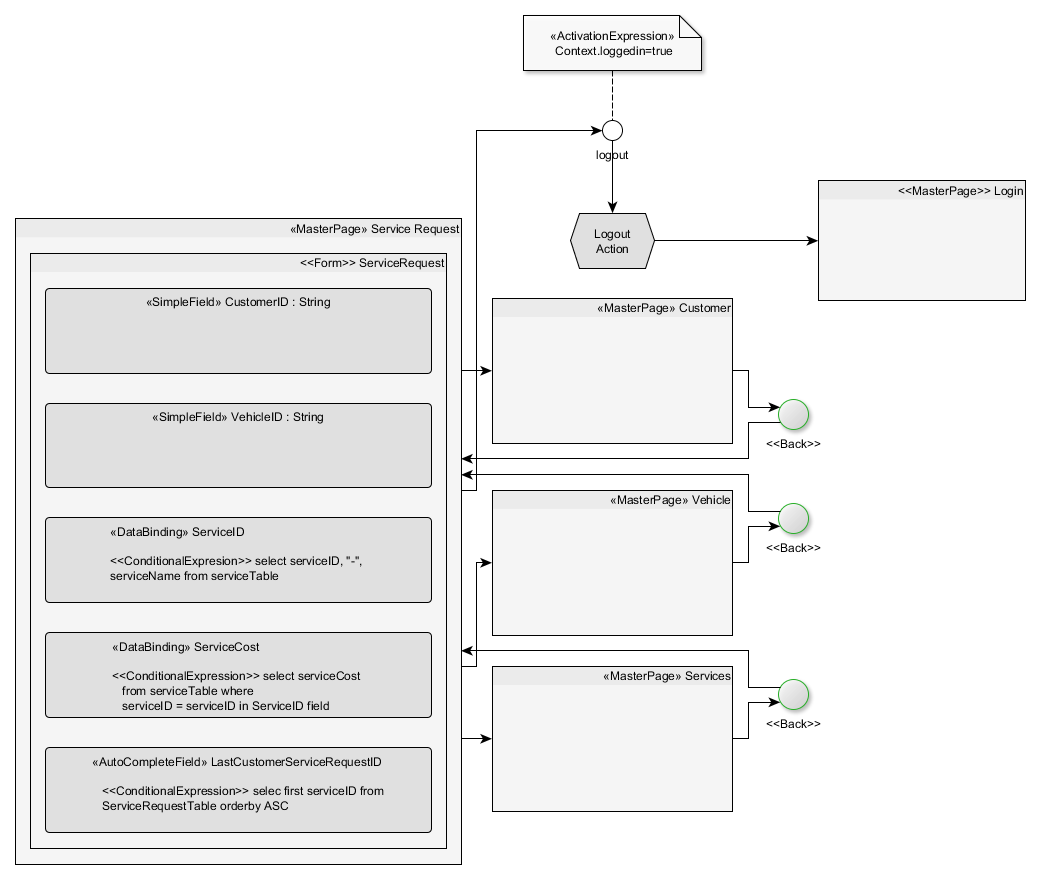
Master Page Login:



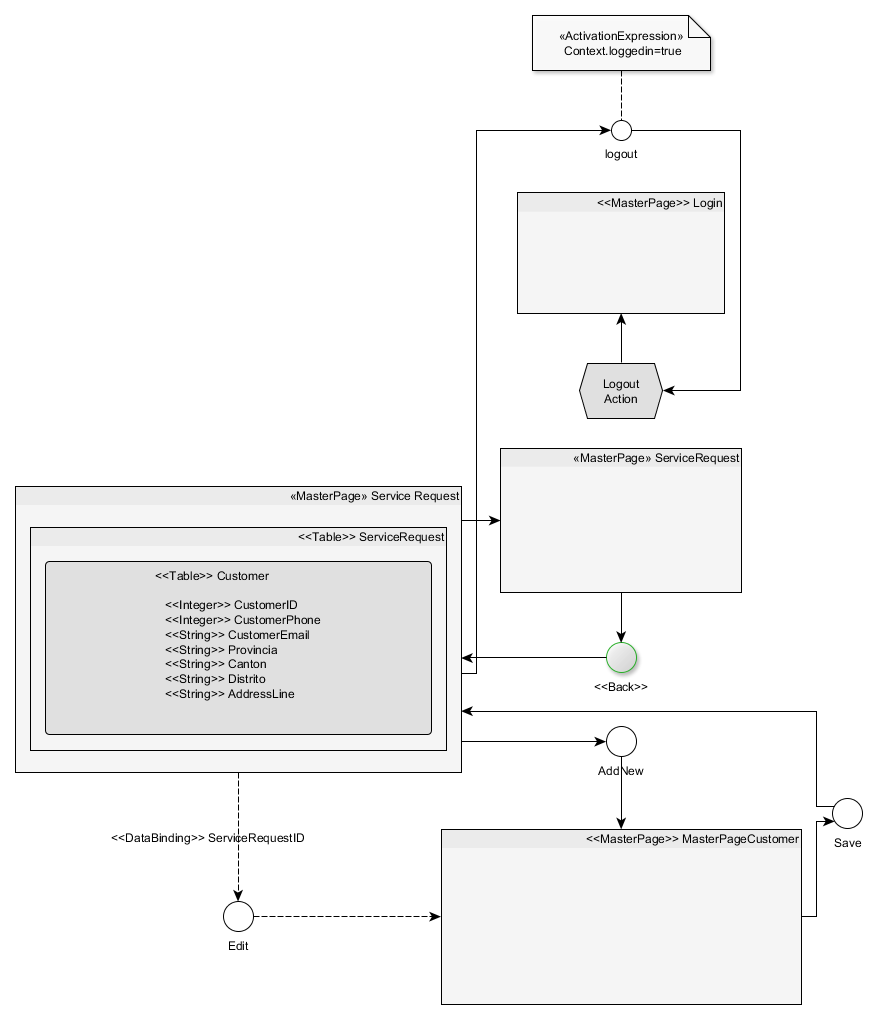
Master Page Service Request View:



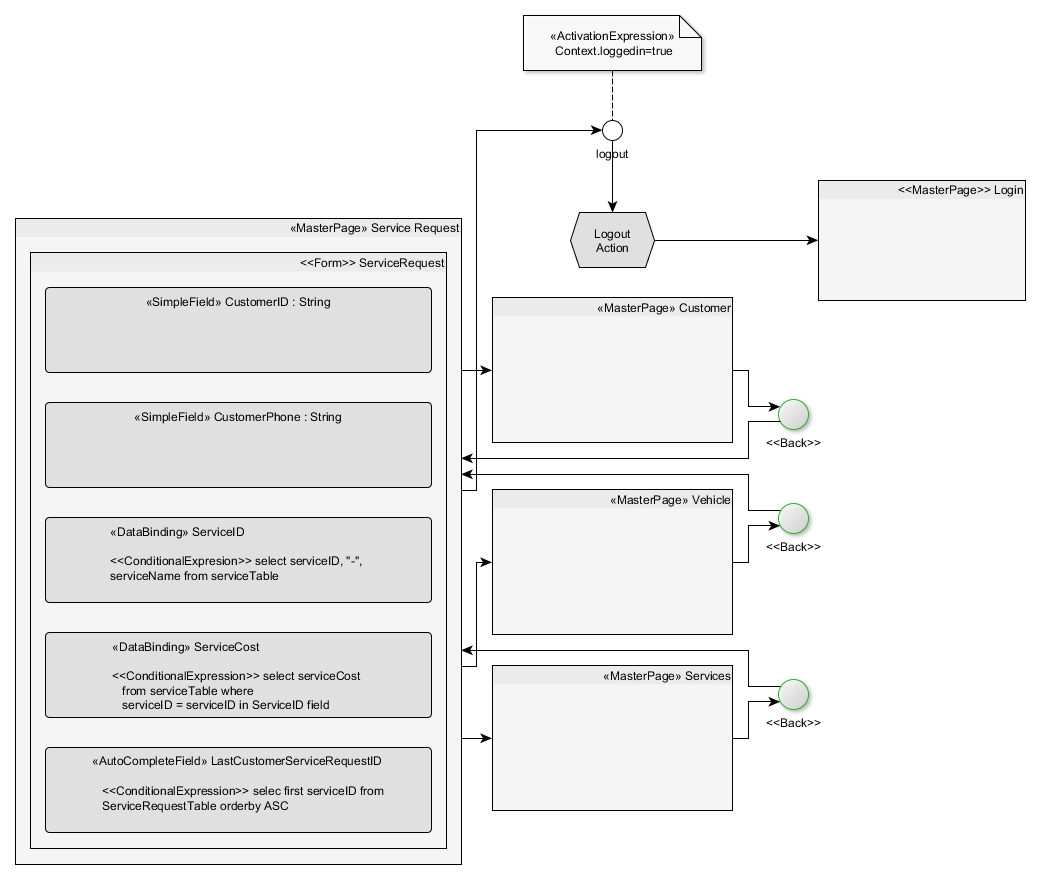
Master Page Service Request Form:



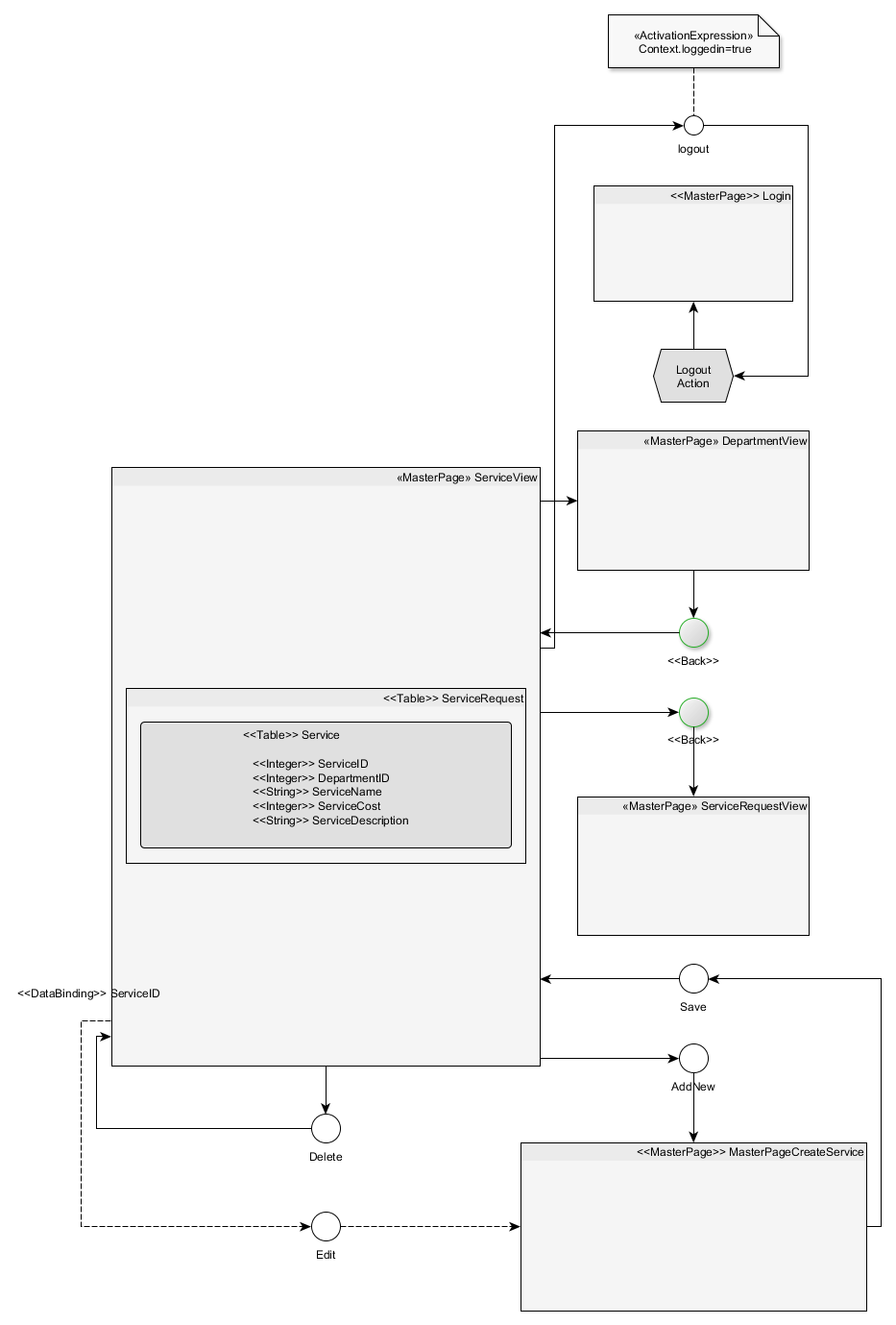
Master Page Customer View:



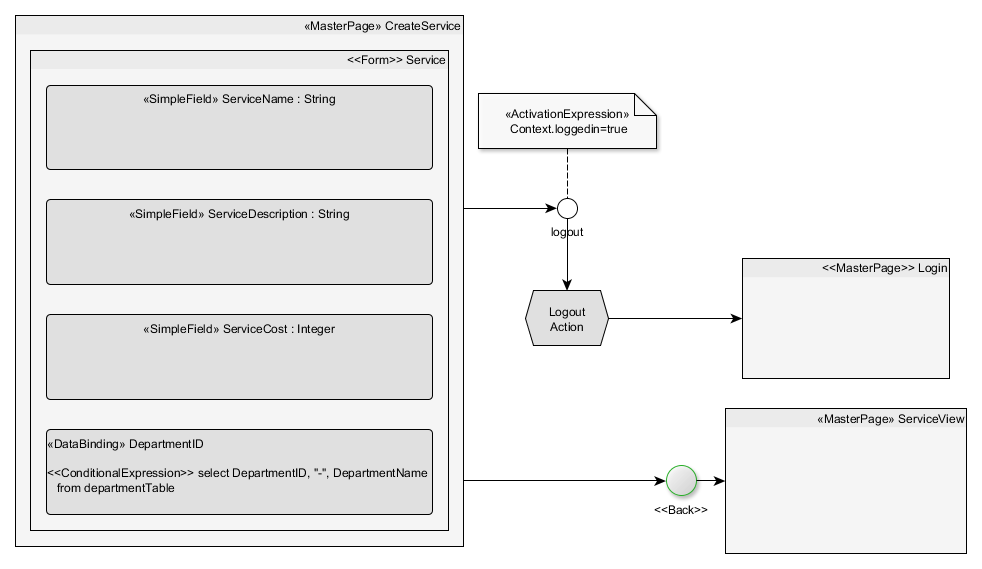
Master Page Customer form:



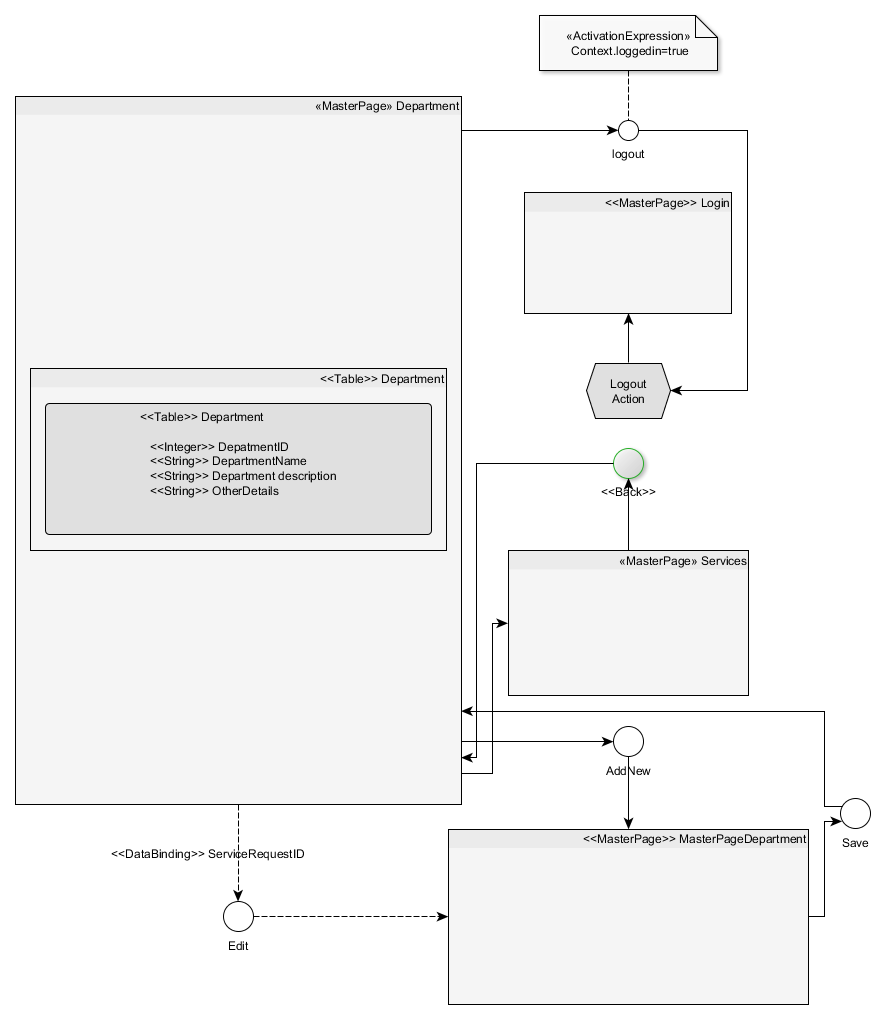
Master Page Service View:



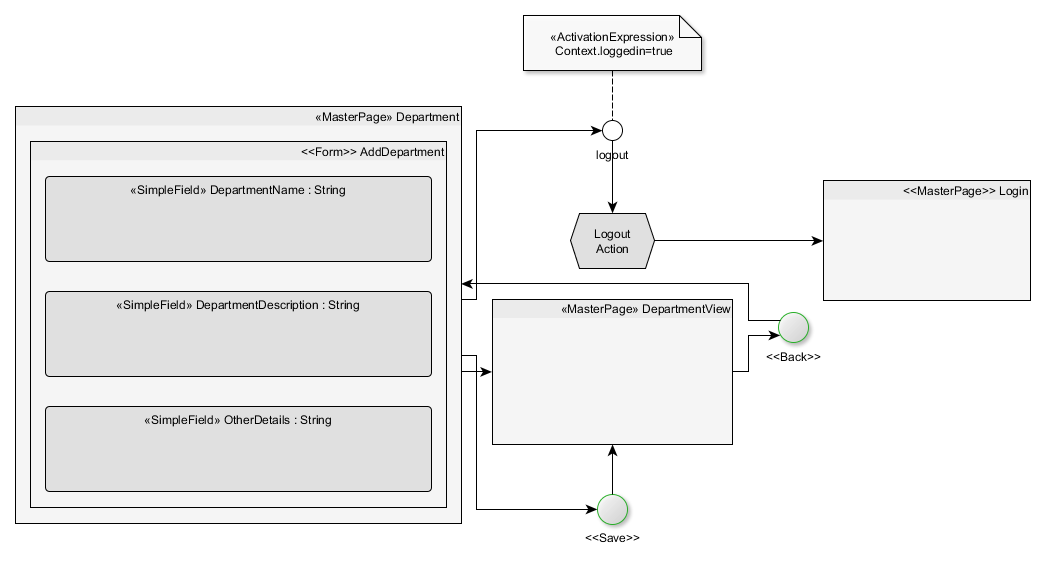
Master Page Service form:



Master Page Department view:



Master Page Department:



Conclusiones

1. Un correcto acuerdo en el diseño previo al desarrollo permite ahorrar tiempo en código, planeación y satisfacción del cliente cuando el producto es finalizado
2. El modelo IFML permite y toma en cuenta muchas de las debilidades de modelos anteriores que programadores reclaman indicando que el modelo no está de acuerdo a lo posible en código
3. El modelo IFML acude al acuerdo entre tanto cliente final como programadores al dar un modelo que en un inicio muestra un diseño visual y después este es trasformado a un modelo de interacciones visual entre los elementos de la arquitectura que es pasado a los desarrolladores
4. IFML se muestra como un modelo muy flexible, pero a su vez esta flexibilidad causa malentendidos en la forma que un grupo de persona envueltas en el proyecto pueden interpretar
5. El modelo IFML aplica mucho texto en el desarrollo de sus diseños lo que a su vez hace más lento el proceso de entendimiento del modelo presentado y hace que el siguiente nivel después del modelado quede poco claro y a su vez confuso en las interpretaciones de lo visual y lo que aparece escrito