Al analizar las reglas de negocio, los requerimientos funcionales y no funcionales, así como los casos de uso y sus respectivos escenarios fue posible realizar una priorización de las características mas importantes que guiaran el desarrollo de la aplicación. Estas características satisfacen a las reglas de negocio, a los requerimientos funcionales y no funcionales. Estas características son:  
  
Fiabilidad: El sistema debe presentar y calcular información confiable que permita a los dueños del negocio tomar decisiones respecto a este. La información presentada por el sistema deber apegarse a las reglas de negocio establecidas. Se indica en los requerimientos:

Seguridad: El sistema debe presentar información solo y únicamente al personal correcto de acuerdo con las funciones que este realiza dentro de la organización. El sistema debe contar con mecanismos que gestionen esta autentificación del personal que opera en el negocio. Se indica en los requerimientos:

Rapidez: El sistema debe asegurar un tiempo de interacción corto para los clientes del negocio que deben interactuar con él, así como para el personal que deberá usarlo para realizar sus actividades diarias. Concretamente el sistema no debe durar mucho tiempo en registrar la asistencia de un cliente, tampoco debe durar mucho tiempo en vender un producto o administrar algún aspecto del negocio: Proveedores, Empleados membresías. Etc.

A continuación, se muestra un diagrama de bloques que describe el funcionamiento abstracto del sistema, así como los módulos que lo componen.

Componente de Registro Entrada/Salida de Clientes

Componente de Administración:

Productos, Proveedores, Empleados, Membresías, Clientes

Componente de Seguridad

Componente de Notificación

Negar/Aceptar Acceso del cliente

Asegurar acceso al personal autorizado

Componente de Ventas

Vender y Actualizar el stock de productos

Notificar al administrador sobre el estado de registro de un cliente

En base a los requerimientos funcionales y no funcionales descritos en el documento de requerimientos se ha definidos algunas de las principales características de la aplicación:

Componente de Registro Entrada/Salida de Clientes

Componente de Administración: Productos, Proveedores, Empleados, Membresías, Clientes

Componente de Seguridad

Componente de Notificación

Negar/Aceptar Acceso del cliente

Asegurar acceso al personal autorizado

Componente de Ventas

Vender y Actualizar el stock de productos

Notificar al administrador sobre el estado de registro de un cliente

Módulo de registro manual

Módulo de registro basado en lectura de código QR

Módulo de Caja registradora

Ventanas emergentes

Identificación por usuario -constreñía

Módulo Administración de productos

Módulo Administración de proveedores

Módulo Administración de Empleados

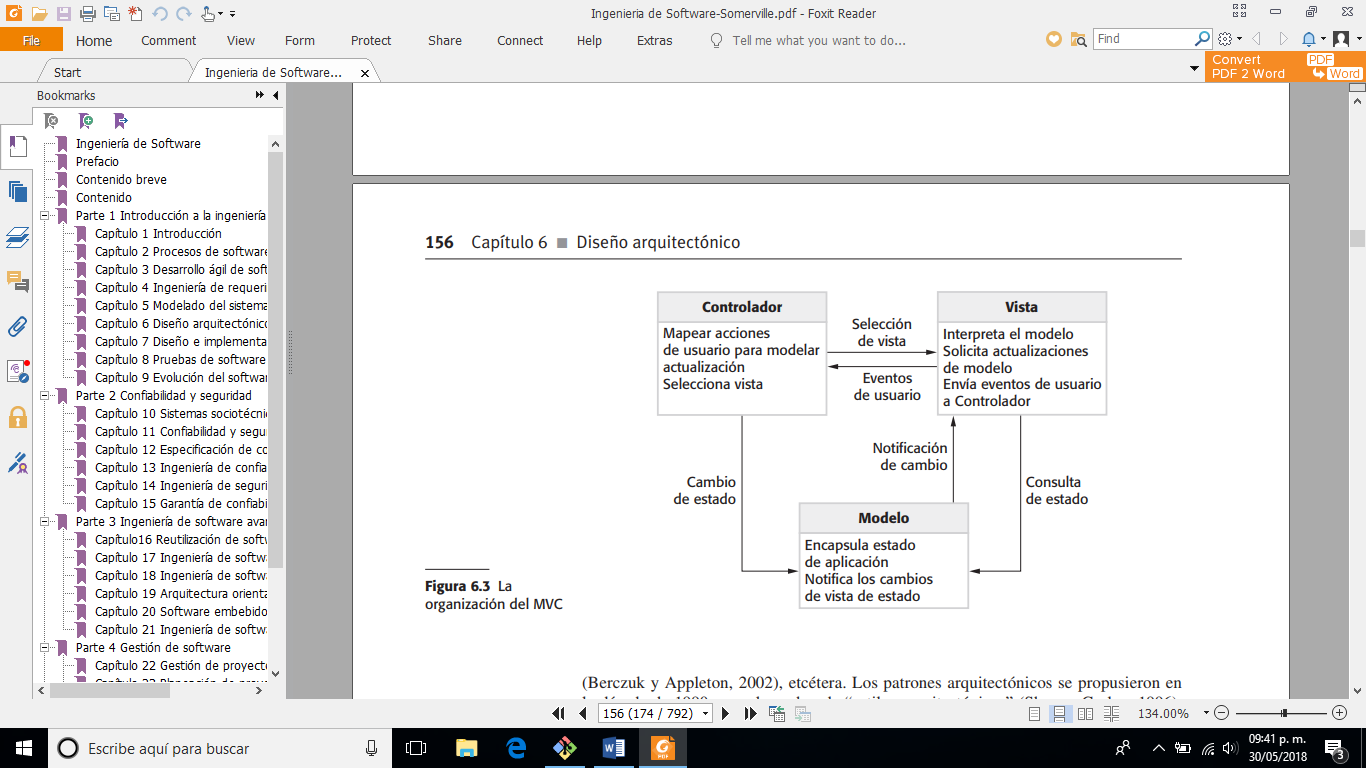
Módulo Administración de Membresías

* La aplicación debe ser una aplicación “*stand alone”*
* La aplicación debe ser una aplicación de escritorio
* La aplicación no debe estar conectada a internet
* La aplicación será construida para sistemas operativos “*Microsoft Windows”*
* La aplicación debe ejecutarse en sistema operativo “Microsoft Windows 7 Ultimate” como mínimo
* La aplicación debe ser construida usando el framework “*.NET*”
* La aplicación debe ser construida usando la versión 3.5 del framework .NET como mínimo
* La aplicación debe guardar información en una base de datos
* El manager de base de datos debe ser MySQL
* La versión mínima del manager de base de datos debe ser la 5.7
* La aplicación debe ser construida usando la tecnología “Windows Form” incluida en el framework .NET   
  La aplicación debe asegurar la confiabilidad de la información
* La aplicación debe asegurar el acceso solo al personal autorizado

A partir de los diagramas de bloques y los requerimientos expuestos anteriormente se ha propuesto una arquitectura que describa el funcionamiento del sistema.

La arquitectura es el nivel de abstracción mas alta que puede tener un sistema de información. Se encarga de modela y representar la interacción entre las diferentes entidades que conforman al sistema de manera general. En una primera instancia es recomendable que se utilicen los llamados “*Patrones arquitectónicos”* estos son soluciones a problemas de desarrollo que ya han sido probadas y comprabas en el contexto de la creación de un sistema información, es decir son plantillas que indican “*como construir”* un sistema de información de manera conceptual basándose en la experiencia de muchos otros proyectos. El patrón arquitectónico elegido para el sistema actual es el bien conocido patrón Modelo, Vista, Controlador (MVC). A continuación, se habla brevemente de este patrón arquitectónico.

|  |  |
| --- | --- |
| Nombre del patrón: Modelo, Vista, Controlador | |
| Descripción | Separa presentación e interacción de los datos del sistema. El sistema se estructura en tres componentes lógicos que interactúan entre sí. El componente Modelo maneja los datos del sistema y las operaciones asociadas a esos datos. El componente Vista define y gestiona cómo se presentan los datos al usuario. El componente Controlador dirige la interacción del usuario (por ejemplo, teclas oprimidas, clics del mouse, etcétera) y pasa estas interacciones a Vista y Modelo |
| Ejemplo | Se usa cuando existen múltiples formas de ver e interactuar con los datos. También se utiliza al desconocerse los requerimientos futuros para la interacción y presentación |
| Uso | Permite que los datos cambien de manera independiente de su representación y viceversa. Soporta en diferentes formas la presentación de los mismos datos, y los cambios en una representación se muestran en todos ellos. |
| Ventajas | Permite que los datos cambien de manera independiente de su representación y viceversa. Soporta en diferentes formas la presentación de los mismos datos, y los cambios en una representación se muestran en todos ellos. |
| Desventajas | Puede implicar código adicional y complejidad de código cuando el modelo de datos y las interacciones son simples. |



A continuación, se presenta la implementación de este patrón abstracto dentro del contacto del desarrollo de la aplicación actual. Note como cada parte de la arquitectura abstracta ahora es reemplazada por una entidad concreta.

**Controlador**   
Sistema de manejo de eventos de Windows forms.  
Vigila las interacciones entre el usuario y la interfaz grafica

**Vista**  
Despliega la información obtenida de las reglas de negocio.   
Archivos \*.cs que describen la disposición de componentes

**Modelo**Representa la información encapsulada en la base de datos y los modelos a los que esta debe apegare según las reglas de negocio de la organización.   
es representado mediante el ORM “*Entity manager*”

Eventos del núcleo .NET

Selección de vista según el rol del usuario

Comunicación a través del ORM “Entity manager”

Consulta de información de la base de datos

Notificaciones de control