Parcial 1 – Seminario de Aplicación Profesional

26/05/2020 – Norte – 5º A

Docentes: Gastón Weingand – Federico Adrián Cetraro

Alumno: Di Modugno Juan Emanuel

1. DER propuesto

A screenshot of a cell phone

Description automatically generated

1. El patrón *Composite* cuya una de sus implementaciones icónicas es el ***Usuario – Patente – Familia*** se utiliza para instrumentar un mecanismo de accesos y permisos en un sistema o aplicación de software.

El punto de similitud respecto de una plataforma como Auth0 es lo ya mencionado respecto de otorgar permisos y accesos a un usuario dado.

El patrón *Composite* solo representa una parte del scope que Auth0 tiene, dado que en sí mismo solo contempla lo inherente a Autorización a través de los incluir a un usuario en una familia, asignarle una patente, o ambas.

En Auth0 se implementa OpenID Connect (OIDC), un protocolo de Autenticación basado en OAuth 2.0, asimismo, como parte de la gestión de Autorización se puede asignar roles y permisos, así como reglas que se pueden administrar desde dicha plataforma.

1. Los elementos que hacen a una correcta definición de un caso de uso son:
   1. El actor que llevará a cabo la ejecución del caso de uso, de ser necesario, puede ser referenciado a través de una cualidad presente en el mismo, usualmente un rol o un permiso. Ej: “Encargado”
   2. Flujo principal, es el listado ordenado de sucesos y eventos que hacen al caso de uso, representa una comunicación bidireccional entre el actor y el sistema, se busca que se estos se expresen de manera lo más detallada posible. Un ejemplo de esto es:
      1. Actor -> Ingresar al listado de artículos
      2. Sistema -> Devuelve el listado de artículos
      3. Actor -> Selecciona uno de ellos y presiona “Modificar”
      4. Sistema -> Valida que el actor tenga permisos, de ser así, muestra el formulario de modificación con los datos del artículo ya cargados.
      5. Actor -> Modifica los valores a voluntad y presiona guardar
      6. Sistema -> Valida integridad de los datos
      7. Actor -> Confirma la operación
      8. Sistema -> Persiste los cambios
   3. Flujo alternativo, representa lo que sucederá en caso de que alguno de los pasos expuestos en el flujo principal no se dé de la manera esperada. Por ejemplo, en el paso ***VI*** puede ocurrir un error en la validación de los datos, eso acarrea el siguiente flujo:
      1. Sistema -> Muestra el/los campos inválidos.
      2. Actor -> Corrige los errores
      3. Sistema -> Valida integridad de los datos
      4. Actor -> Confirma la operación
      5. Sistema -> Persiste los cambios
   4. Pre-Condición, es el estado del sistema antes de la ejecución de un caso de uso, si bien está expresa como singular, puede detallarse una lista de condiciones para facilitar el contraste con la Post-Condición.
   5. Post-Condición, es el estado del sistema después de la ejecución de un caso de uso, de manera análoga a la Pre-Condición, puede detallarse un listado de condiciones.
2. Los modelos prescriptivos proponen un conjunto de elementos distintos entre sí para llevar a cabo el desarrollo de software.

Los elementos mencionados previamente son actividades, acciones, tareas, flujos de trabajo, fundamentos y productos, dichos elementos facilitan la obtención de software de alta calidad.

Existen varios tipos, a continuación se los nombra a modo de presentación:

* + Cascada
  + En V
  + Proceso Incremental
  + Evolutivo
    - Prototipos
    - Espiral

1. El Proceso Unificado (UP) pertenece a los modelos prescriptivos de carácter incremental e iterativo, se centra en la arquitectura y es dirigido por casos de uso.
2. Las disciplinas comprendidas dentro del UP contemplan:
   1. La extracción de requerimientos
   2. El análisis y diseño
   3. La implementación
   4. El testing
   5. El despliegue

Las disciplinas arriba mencionadas tienen su correlación con etapas dentro del UP:

* Inicio 🡪 Extracción de requerimientos para generar un modelo primario
* Elaboración 🡪 Creación de modelos más refinados con un enfoque en clases y representaciones a nivel arquitectónico
* Construcción 🡪 Refina y traduce el diseño a componentes de Software. En esta etapa pueden realizarse pruebas unitarias y de integración
* Transición 🡪 Enfoca los esfuerzos en realizar pruebas de aceptación con el usuario final y prepara los últimos detalles previo a la instancia productiva
* Producción 🡪 Despliegue, monitoreo y soporte

1. La arquitectura de software involucra las decisiones y concepciones que sirven como plataforma para la construcción de un sistema. En ella se define los componentes a alto nivel y su interacción entre sí, eso posibilita ver de antemano los potenciales problemas que puedan surgir y plantear estrategias para resolverlos. Hay modelos de arquitectura que han demostrado ser funcionales para situaciones determinadas, su complejidad varía pero a la hora de evaluar el costo-beneficio, resultan atractivas igualmente.
2. A la hora de hablar del producto del software tenemos que comprender que no solo hablamos del programa/sistema/ejecutable que será entregado, la documentación generada mediante la extracción de requerimientos, los modelos preliminares, los diagramas de dominio, clases, secuencia, etc… también conforman el producto final del software. Los mismos deberán ser revisitados cada vez que el sistema sufra una modificación, esto permitirá tener trazabilidad y llevar a cabo una correcta gestión del cambio.
3. Al referirnos a los diagramas de comportamiento que plantea UML podemos mencionar los siguientes:
   1. Diagrama de Actividad:
      1. *Se utiliza para modelar una secuencia de acciones y condiciones tomadas en un proceso dado.*
      2. *Representa flujos de trabajo paso a paso.*
      3. *Sus componentes son: Actividad, Acción, Flujo de Control, Nodo de inicio, Nodo de Fin, Nodos de Decisión y Combinación. Nodos de Bifurcación y Unión.*
   2. Diagrama de Interacción
      1. Se utiliza para modelar aspectos dinámicos de un sistema.
      2. Es un grupo que comprende una serie de subdiagramas que representan detalles específicos.
      3. Subdiagramas:
         1. Diagrama de Comunicación
            1. *Modela las interacciones entre objetos o partes en términos de mensajes secuenciados.*
            2. *Es una versión simplificada del diagrama de colaboración.*
            3. *Sus componentes son: Objetos, Mensajes y Actores.*
         2. Diagrama de Secuencia
            1. Muestra las comunicaciones entre objetos y los mensajes involucrados en tales flujos.
            2. No muestra lógica de procedimiento complejo.
            3. Sus componentes son: Línea de vida de un objeto, Activación y Mensaje.
         3. Diagrama Global de Interacción
            1. Es una representación gráfica del diagrama de secuencia o el de comunicación junto al de actividad.
         4. Diagramas de Tiempo
            1. Muestra las interacciones de un diagrama para analizar el tiempo.
            2. Muestra los cambios que ocurren en las líneas de vida a lo largo de una línea de tiempo.
            3. Sus componentes son: Línea de vida de estado y línea de vida de valor.
   3. Diagrama de Máquina de Estados
      1. Representa los posibles estados de una entidad, clase u objeto. Los estados expresados dentro de tal diagrama serán los que pueda tomar la ya mencionada entidad.
      2. El diagrama muestra las transiciones, estas representan las causas, circunstancias y efectos de cambio entre dos estados.
   4. Diagrama de Casos de Uso
      1. Representa los distintos casos de uso que existirán en el sistema, estos pueden agruparse basados en su alcance o contexto.
      2. Muestra la relación entre distintos casos de uso, esta puede ser una relación de inclusión, de extensión o de generalización.
      3. Sus componentes son: Actor, Relación y Caso de Uso.
4. Como ya se mencionó en el punto 6, existen etapas claramente delimitadas en el proceso de software, estas se relacionan a las actividades estructurales de dicho proceso. A continuación se enumera a tales actividades:
   1. Comunicación
   2. Planificación
   3. Modelado
   4. Construcción
   5. Despliegue
5. El desarrollo ágil permite iterar de manera más rápida sobre el software, tiene una mirada más enfocada en el producto en su concepción total de manera que los stakeholders están más cerca del proceso de implementación y pueden dar feedback de manera más continua. Los tiempos entre cada hito se acortan, esto favorece la generación de entregables con mayor cadencia. Es la instrumentación de metodologías ágiles al mundo del software, existen muchos tipos basados en la necesidad, la metodología más utilizada es la de Scrum y acarrea cambios culturales en un equipo, se trabaja en plazos de tiempo limitados denominados Sprint, al final de cada uno de estos se genera un entregable, puede ser un cambio parcial o generar una nueva versión dependiendo la duración de tal sprint, claro está que también se relaciona con la complejidad del sistema a modificar.
6. Al expresar una comparativa entre las clases Hashtable y Dictionary provistas por C# podemos compararlas en 4 aspectos técnicos y uno a nivel organizacional.

|  |  |  |
| --- | --- | --- |
| Aspecto | Hashtable | Dictionary |
| Namespace | Se incluye en *System.Collections* | Se incluye en *System.Collections.Generic* |
| Tipado | No utiliza generics, eso quiere decir que puede guardar conjuntos de clave-valor de cualquier tipo. | Es una colección genérica, puede guardar conjuntos de clave-valor de tipos específicos. |
| Multithread | Aplica estrategia de protección para ejecución multithread. | Solo los miembros que son públicos y estáticos están protegidos en ejecución multithread. |
| Valor inexistente | Por defecto devuelve un valor null al no encontrar una clave en la colección. | Al no encontrar una clave en la colección lanza una excepción. |
| Velocidad de devolución de datos | Es lenta debido a que debe convertir los datos almacenados en al ser devueltos. | Es más rápida que la Hashtable. |

Práctico

Diagrama de clases (Domain)

A picture containing screenshot

Description automatically generated

*Deposit y Withdrawal podrían ser una única clase o subclases de Movimiento, habría que adicionar un campo con un multiplicador para llevar a cabo la operatoria y los cálculos de balance.*