



**TALLER 1 /** Evaluación Unidad N°2

|  |  |
| --- | --- |
| **Nombre de la Unidad:** | **ASP.NET y Sitios web** |
| **Nombre de la Asignatura:** | **Programación I** |

# Aprendizaje Esperado

En esta actividad lograrás adquirir conocimientos necesarios para construir una página web con ASP.NET y C# capaz de administrar datos almacenados en memoria por medio de formularios web.

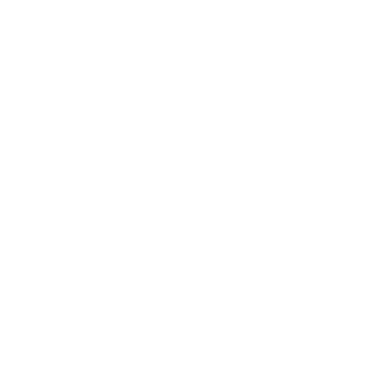
## INSTRUCCIONES GENERALES

Una vez realizada la lectura comprensiva de la unidad I y II del material de estudio, es fundamental la realización de este taller, el que tiene por objetivo medir la correcta internalización y aplicación de los conceptos abordados en las mencionadas unidades.

El desarrollo de este trabajo es individual, no se permitirán entregas de talleres en duplas o en grupos

## CONTENIDOS ASOCIADOS AL TALLER:

* Arquitectura n-capas
* ASP.NET
* Formularios Web Forms
* Controles ASP.NET

****

**Esta actividad se complementa con una pauta de autoevaluación al finalizar, para que verifiques el logro**

**de tus aprendizajes**

**INSTRUCCIONES ESPECÍFICAS**

El siguiente caso será utilizado en la presente unidad 2 y en el examen de la presente asignatura, en donde al finalizar esta, deberás ser capaz de disponibilizar un sitio web por medio de IIS que permita administrar registros almacenados en memoria por medio de formularios web funcionales que cumplan con los requerimientos del cliente.

Específicamente durante esta segunda unidad, usted deberá ser capaz de construir un sitio web utilizando el framework de ASP.NET con C# como lenguaje de programación para administrar registros almacenados en memoria por medio de formularios web y separando responsabilidades en el software.

Los puntos para desarrollar serán los siguientes

**Se solicita:**

|  |
| --- |
| Realice la/las siguientes actividades:   1. Utilizar aplicación ASP.NET para desarrollar la interfaz de usuario del proyecto 2. Utilizar C# como lenguaje de programación para desarrollar el proyecto 3. Crear las capas de datos y modelo/negocio para separar roles dentro del   proyecto   1. Crear Master Page como diseño general del proyecto 2. Crear menú de navegación para el sitio 3. Utilizar formularios Web Forms para presentar al usuario las funcionalidades del   sitio web   1. Utilizar los controles necesarios modificando sus propiedades y ejecutando 2. los eventos necesarios |

**CASO DE ESTUDIO**

La empresa PeCe’s es una de las tiendas de ventas de productos tecnológicos con mejores ventas en los últimos años en el Gran Concepción. Debido a esto, el dueño de la tienda ha decidido abrir un taller de reparación de este tipo de productos para ampliar la cantidad de servicios que ya disponen. Para esto, le ha solicitado a usted como desarrollador de software crear un sitio web con ASP.NET montado en el servidor web de IIS que sea capaz de administrar los tickets de soporte que vayan llegando a la tienda, ya sean de personas naturales o de empresas.

Este proyecto se dividirá en dos etapas:

1. Construcción sitio web con ASP.NET y C#
2. Implementación de validaciones y disponibilizar sitio web por medio de IIS.

Usted deberá desarrollar la primera etapa, es decir, crear sitio web con ASP.NET utilizando formularios Web Forms, Master Page, controles de distinto tipo y separando las capas de datos y modelo, agregando las referencias necesarias entre cada uno de los proyectos.

Para que esta primera etapa sea aprobada, se deberán cumplir con los siguientes items:

**A NIVEL GENERAL:**

* El sitio web debe ser construido con ASP.NET como Framework Frontend y C# como lenguaje de programación
* Debe existir separación de roles por medio de capas
* La capa de datos solo debe estar referenciada por la capa de modelo/negocio
* La capa de modelo/negocio solo debe estar referenciada por la capa de interfaz de usuario (ASP.NET)
* Las capas de datos y modelo/negocio deben ser proyecto de tipo Biblioteca de Clases (Class Library)

**A NIVEL DE LA CAPA DE DATOS**

* Se deben crear las siguientes clases de tipo Entity con las siguientes estructuras:
  + **ClienteEntity**
    - Propiedades
      * public string Id { get; set; } = Guid.NewGuid().ToString();
      * public string Nombre { get; set; }
      * public string Rut { get; set; }

public string Telefono { get; set; }

* + - * public string Email { get; set; }
  + **PersonaNaturalEntity**
    - Debe heredar sus propiedad de la clase ClienteEntity
  + **EmpresaEntity**
    - Debe heredar sus propiedad de la clase ClienteEntity
    - Adicionalmente, se debe agregar la propiedad
      * public string RazonSocial { get; set; }
  + **TicketEntity**
    - Propiedades
      * public string Id { get; set; } = Guid.NewGuid().ToString();
      * public ClienteEntity Cliente { get; set; }

public string Producto { get; set; }

* + - * public string Descripción { get; set; }
      * public string Estado { get; set; }
      * private DateTime \_createdAt { get; set; } = DateTime.Now;
    - Algunas consideraciones adicionales:
      * Se debe crear método público que retorne la propiedad \_createdAt
  + **TicketEntityCollection**
    - Debe ser una clase estática
    - Se deben crear dos propiedades estáticas, una privada y otra pública
    - La propiedad estática privada debe ser de tipo List<TicketEntity> llamada \_listadoTickets
    - La propiedad estática pública debe ser de tipo List<TicketEntity> llamado ListadoTickets. Esta propiedad debe tener asignados sus métodos get y set. En el método get se debe retornar la propiedad privada. En el método set se debe asignar el valor de la propiedad privada. El resultado debe ser como el de la imagen.

A computer screen with text

Description automatically generated

**A NIVEL DE LA CAPA DE MODELO/NEGOCIO**

* Se deben crear las siguientes clases:
  + **Cliente**
    - Debe tener la misma estructura que la clase ClienteEntity, pero para la propiedad Id ya no será necesario asignar un Guid.
  + **PersonaNatural**
    - Debe heredar sus propiedades desde la clase Cliente
  + **Empresa**
    - Debe heredar sus propiedades desde la clase Cliente
    - Adicionalmente, se debe agregar la propiedad
      * public string RazonSocial { get; set; }
  + **Ticket**
    - Debe tener la misma estructura que la clase TicketEntity, pero para la propiedad Id ya no será necesario asignar un Guid.
    - Adicionalmente, la propiedad ClienteEntity debe ser de tipo Cliente
  + **TicketController**
    - Debe ser una clase estática, y por ende, los método también
    - Métodos
      * **Create**
        + Debe recibir un objeto de tipo Ticket como parámetro de entrada
        + En base al tipo de cliente creado, se deberá crear el objeto EmpresaEntity o PersonaNaturalEntity según corresponda
        + Se deben realizar las transformaciones necesarias de clases de capa de negocio a clases de capa de datos.
        + Debe agregar el objeto TicketEntity a la lista de la capa de datos
        + El método debe retornar un mensaje de éxito si se ha creado el registro
      * **Read**
        + Debe recibir el id del ticket a buscar
        + Se debe utilizar método Find para buscar en la lista para buscar registro en base al id
        + En caso de no existir un ticket con el id proporcionado, retornar null
        + En base al tipo de cliente obtenido, se deberá crear el objeto EmpresaEntity o PersonaNaturalEntity según corresponda
        + Se deben realizar las transformaciones necesarias de clases de capa de negocio a clases de capa de datos.
        + Se debe retornar el ticket encontrado
      * **Update**
        + Se deben recibir el id del ticket, el producto, descripción del producto y el estado de revisión del ticket; el email y teléfono del cliente.
        + Se debe utilizar método Find para buscar registro en la lista en base al id
        + Si no se encuentra registro en base al id, retornar mensaje de error
        + Se deben actualizar los valores de las propiedades que correspondan al ticket encontrado. Ojo, esto incluye a las propiedades del cliente
        + Se debe retornar un mensaje de éxito si se ha logrado actualizar el registro
      * **Delete**
        + Se debe recibir el id del ticket a eliminar.
        + Se debe utilizar método Find para buscar registro en la lista en base al id
        + Si no se encuentra registro con el id proporcionado, retornar mensaje de error
        + Se debe utilizar el método Remove para eliminar el registro.
        + Se debe retornar mensaje de éxito si se ha logrado eliminar el registro
      * **ReadAll**
        + Debe retornar un listado de objetos de tipo Ticket
        + Por cada uno de los registros presentes en el listado de la capa de datos, se deben realizar las transformaciones de clase tipo Entity a clase de capa de modelo/negocio.
        + En base al tipo de cliente asociado a un ticket, se deberá crear el objeto Empresa o PersonaNatural según corresponda
      * **Search**
        + Se debe recibir una cadena de texto como parámetro de entrada
        + Se debe filtrar por nombre del cliente, rut del cliente, o estado de la reparación.
        + Por cada uno de los registros filtrados se deben realizar las transformaciones de clase tipo Entity a clase de capa de modelo/negocio.
        + En base al tipo de cliente asociado a cada uno de los tickets filtrados, se deberá crear el objeto Empresa o PersonaNatural según corresponda
        + Debe retornar un listado de objetos de tipo Ticket

**A NIVEL DEL PROYECTO DE ASP.NET**

* **Para la página maestra**
  + Se debe crear una página maestra llamada “MPSitio”, la cual deberá ser utilizada por todos los formularios Web Forms a crear. Se debe considerar lo siguiente para la página maestra:
  + Debe tener el logo de la empresa (es compartido más abajo)
  + Debe contener el menú de navegación
  + Además, debe contener un formulario de búsqueda avanzada con los controles correspondientes.
* **En cuanto al menú de navegación**
  + Debe contemplar el origen de datos y controles necesarios para la correcta navegación en el sitio.
  + Solo debe tener tres opciones de navegación: Home, Agregar Ticket y Listar todos
* **En cuanto a los formularios Web Forms**
  + Debe existir una página de bienvenida. El nombre debe ser Home, mostrando el texto “Bienvenido! Ingresa un ticket de soporte para empezar!”
  + Las siguientes acciones administrativas deben tener su propio formulario Web Forms:
    - Creación de ticket
    - Detalle de ticket
    - Actualización de ticket
    - Listado de tickets
* **Para el formulario de creación de ticket**
  + Por defecto, debe permitir ingresar los valores de la clase Cliente y de la clase Ticket. Cada campo para ingresar debe tener un control de tipo etiqueta que indique se debe ingresar y el control necesario para ingresar o asignar el valor a la propiedad de la clase Cliente
  + Además, deberá existir un control de tipo drop down list que permita elegir el tipo de cliente: Persona natural o Empresa. En base a esto, será el objeto para crear y asignar al ticket de soporte. En caso de ser presionada la opción “Empresa”, el campo para ingresar la razón social deberá ser habilitado.
  + El formulario deberá ser enviado por medio de un botón.
  + Al ser agregado el ticket, se deberá redireccionar a la página de listado de todos los registros, mostrando el mensaje “El ticket fue agregado con éxito”. De lo contrario, “El ticket no pudo ser agregado”
  + El mensaje deberá ser pasado por medio de la url
* **Para el formulario de listado de tickets**
  + Deberá mostrar una grilla de datos, cuyo origen de datos sea el método controlador que lista todos los registros.
  + En caso de que no existan registros, se deberá mostrar un mensaje que diga “No existen registros disponibles para mostrar”. Esto debe esconder la grilla de datos, dejándola invisible para el usuario.
  + La grilla de datos debe contener una sola columna por defecto, la cual permita redireccionar al formulario de detalle pasando el id del registro por la url. El botón debe tener el texto “Ver detalle”
* **Para el formulario de detalle**
  + Se deberá obtener el id del ticket a mostrar en detalle desde la url
  + Además, deberán existir dos botones: uno que permita volver al listado de todos los tickets y que debe mostrar el texto “Volver”, y otro que redireccione al formulario de actualización del registro y debe mostrar el texto “Actualizar”. El id del ticket a actualizar deberá ser pasado por url.
  + En caso de no existir un registro, se debe mostrar un mensaje de error.
* **Para el formulario de actualización**
  + Se debe obtener el id del ticket a actualizar desde la url y los datos deben ser cargados en los controles que corresponden al ingresar a la página.
  + En cuanto a la información del cliente, solo se puede actualizar el teléfono e email.
  + En cuanto a información del ticket, solo los campos producto y descripción podrán ser editables
  + Los otros campos también deberán ser mostrados pero no pueden ser editables.
* **Para el formulario de búsqueda avanzada**
  + Al enviar el criterio o filtro de búsqueda avanzada ingresado en dicho formulario, se deberá redireccionar a la página de listado de tickets, en donde solo se deberán listar los registros que calcen con el filtro.
  + El filtro deberá ser pasado por url y se deberá evaluar si existe filtro, entonces listar en base al filtro, si no, listar todos.
  + Los filtros establecidos en primera instancia (primera etapa) se deben mantener.
  + Se debe pasar el listado ya filtrado como origen de datos para la grilla de datos
  + Si no existen registros que calcen con el filtro, se deberá mostrar un mensaje de error y esconder la grilla de datos.
* **Algunas consideraciones adicionales:**
  + Todos los controles deben tener un nombre único y definido en base a los estándares explicados.
  + Todos los controles Drop Down List deben mostrar como por defecto la opción “Seleccionar”, la cual no puede ser seleccionada por el usuario, es decir, debe estar deshabilitada.
  + El formulario de búsqueda avanzada dispuesto en el Master Page, deberá tener los controles necesarios (Label, Button, TextBox).
  + Los formularios de ingreso, actualización y detalle de datos deberán estar ordenados dentro de una tabla HTML.
* **Logo**

A computer and computer with fish and tools

Description automatically generated

**Información Importante**

Esta actividad será evaluada por el profesor de la asignatura, por lo que se sugiere, para su realización, considerar los Criterios de Evaluación detallados al final de este documento.

El puntaje total de este taller es de 100 puntos, y el puntaje mínimo de aprobación es de 60 puntos.

**FORMALIDADES**

* Se debe utilizar Visual Studio Community 2022 o Visual Studio Code para desarrollar la aplicación
* Se debe utilizar C# como lenguaje de programación
* El proyecto de entrega del proyecto debe ser en formato .zip
* El archivo debe ser subido a plataforma.

**Pauta de Evaluación**

|  |  |  |
| --- | --- | --- |
| **Criterios de Evaluación** | **Indicadores de Logro** | **Puntaje** |
| La arquitectura n-capas se distingue como forma de orden y jerarquización de responsabilidades dentro del desarrollo de una aplicación | Se utilizan bibliotecas de clase para generar las capas de datos y modelo o negocio dentro del proyecto, generando las referencias necesarias entre proyectos | 10 |
| La administración de datos por medio de colecciones de tipo lista y/o arraylist, y los métodos necesarios para dicho propósito son implementados según requerimientos | En la capa de datos se crea la colección estática que almacenará los registros durante la ejecución de la página | 10 |
| En la capa de negocio o modelo se crean las clases modelo y controladoras necesarias | 20 |
| Construye el sitio web con ASP.NET, utilizando web forms y master page como estructuras generales del sitio web | Utiliza master page para construir plantilla general dentro del proyecto | 10 |
| Utiliza formularios web forms para generar las diversas pantallas de administración solicitadas | 20 |
| Los controles ASP.NET necesarios para generar la interfaz de usuario son utilizados modificando propiedades y comportamientos según requerimientos | Utiliza controles nativos de asp.net según el tipo de dato a ingresar y los nombra según las reglas especificadas | 15 |
| Modifica las propiedades a nivel de interfaz y ejecuta eventos de para los controles según los requerimientos | 15 |
| Puntaje Total | | 100 |

**Pauta de Autoevaluación:**

A continuación, te invitamos a responder una pauta de autoevaluación, para que verifiques lo aprendido en esta actividad y el logro del o los objetivos planteados inicialmente. Recuerda, en caso de presentar dudas, debes participar de forma permanente en los foros, donde el docente te retroalimentará y dará respuesta a tus inquietudes.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Criterios de revisión del informe con la información recopilada:** | **Completamente logrado**  **(4)** | **Medianamente logrado**  **(3)** | **Parcialmente logrado**  **(2)** | **No logrado**  **(0)** |
| 1. Logré asociar correctamente los conceptos de xxxxxxx con xxxxxxx | 4 |  |  |  |
| 1. Indiqué correctamente los principios de xxxxxxxxxxxxx | 4 |  |  |  |
| 1. Elabora un análisis asociado al texto de xxxxxxxxxxx |  |  |  | 0 |
| 1. Finalicé la actividad, comprendiendo cada uno de los ejercicios desarrollados | 4 |  |  |  |
| 1. Logré el objetivo planteado para esta actividad |  | 3 |  |  |
| **Puntaje Total** | **20 puntos** | | | |
| **Puntaje Obtenido** | **XX Puntos obtenidos** | | | |

