Sistema de Cursos Online

Programación Web III

Departamento de Ingeniería e Investigaciones Tecnológicas

2013 – 2C

CONTENIDO

RESUMEN PROYECTO

Propósito y Objetivos

Crear una web simple que proporcione la funcionalidad para que un profesor pueda administrar sus cursos y exámenes. Y que a su vez los alumnos puedan ingresar y completar los mismos, recibiendo su calificación de manera automática.

Referencias de sitios que poseen funcionalidades similares:

- https://www.engrade.com/
- http://www.schoology.com/

Carga de Datos Inicial

Al momento de la entrega deberá al menos existir 1 Profesor, con 1 curso.
El curso del profesor deberá tener:

- 4 alumnos
- 1 Examen “Primer Parcial” tarea con fecha tope ya vencida
    - El mismo deberá tener 2 alumnos que hayan realizado el examen (1 aprobado y 1 desaprobado).
- 1 Examen “Parcial en clase” tarea con fecha tope no vencida, duración 15 minutos, porcentaje para aprobación 60%.
    - **El examen deberá de tener 5 preguntas referidas a los temas vistos durante la cursada**
        - **Al menos 1 pregunta donde para que la respuesta correcta sean 2 o más.**
    - **TODOS alumnos del grupo que realizaron el tp deben tener este examen como aprobado.**

Se deberá tener disponible al momento de entrega los datos de login del Profesor y de un alumno de su curso.

En esta sección se detallarán los detalles funcionales para cada componente de la solución.

1. Login de Profesor

Corresponde a la primera acción que hará un usuario sobre el sitio a fin de poder utilizar sus servicios.

1. Campos del formulario de Login:
    a. *Mail
    b. *Contraseña

    *campos obligatorios
2. Con el mail y la contraseña el profesor podrá ingresar al sitio.
3. Al ingresar al sistema un Profesor tendrá 3 links disponibles en su menú
    a. Mis Cursos (2. ABM de Cursos)
    b. Mis Exámenes (4. Administración de Exámenes)
    c. Salir (vuelve a la página de login)

2. ABM de Cursos (Solo Profesor)

El Profesor podrá administrar solo sus cursos. Para ello debe de tener una grilla con sus cursos y un link/botón de creación.

1. Campos de Curso:
    a. *Nombre
    b. *Activo - checkbox por defecto en Verdadero
    c. Fecha Inicio (1) (2)
    d. Fecha Fin (1) (3)
    e. Email Alumnos – un textbox multilinea donde se ingresen emails separados por enters o ; (lo que sea más fácil para el desarrollador)

    *campos obligatorios

    (1) Debe de crearse un UserControl (archivo de extensión .ascx) y utilizarse en todos los campos para elegir una fecha. Utilizar algún componente de para elegir fechas (por ejemplo: http://jqueryui.com/datepicker/).

    (2) Por defecto se deberá autocompletar con la fecha actual

    (3) Por defecto se deberá autocompletar con la fecha actual + 1 mes. (Ejemplo: Inicio 30/09/2013, Fin 30/10/2013)
2. El Profesor podrá crear/editar/borrar cursos.
3. Al momento de crear, se creará un alumno por cada email ingresado (en caso de ya existir, no se crea) y se le agregará el curso. Por defecto le asignará a los alumnos creados una contraseña igual a su mail.
4. Al momento de editar, se podrá editar la lista de emails.
    a. En caso de que el alumno no exista en el sistema, se creará y se le agregará el curso.
    b. En caso de que el alumno exista y no este asignado al curso, se le asignará este curso (sin quitarle los otros cursos que tenga asignado).
    c. En caso de que al modificar la lista, se haya quitado algún email, debe removerse a ese alumno del curso (pero no eliminarse al alumno del sistema).
5. Solo se podrán borrar aquellos cursos que no posean exámenes o alumnos asignados
6. La grilla de Cursos deberá de poseer lo siguiente:
    a. Nombre
    b. Activo
    c. Cantidad de Alumnos
    d. Fecha Inicio
    e. Fecha Fin
    f. Link/botón de edición
    g. Link/botón de borrado

3. Login de Alumno

Corresponde a la primera acción que hará un alumno sobre el sitio a fin de poder utilizar sus servicios. Previamente el alumno debió haber sido creado por algún Profesor al asignarlo a un curso (luego puede ser desasignado de un curso, pero el usuario no se borra)

1. Con el mail y contraseña podrá ingresar al sitio:
    a. *Email
    b. *Contraseña
    *campos obligatorios
2. Cada usuario podrá editar sus datos personales incluyendo su contraseña, pero no su Email.
3. Campos posibles de usuario
    a. DNI
    b. Nombre
    c. Apellido
    d. Email (deshabilitado)
    e. (4) Contraseña Actual
    f. (4) Contraseña Nueva
    g. (4) Repetir Contraseña Nueva
    (4) Solo se deben ingresarse las contraseñas en caso de querer modificarse la actual, caso contrario no es requerida.
4. Al ingresar al sistema un alumno tendrá 2 links disponibles en su menú
    a. Editar mis Datos.
    b. Exámenes Disponibles (5. Lista de exámenes)
    c. Salir (vuelve a la página de login)

4. Administración de Exámenes (para Profesor)

El Profesor podrá crear Exámenes para sus cursos. Los mismos poseerán:

1. *Curso – Combo o lista de sus cursos donde deberá elegir uno.
2. *Nombre.
3. Descripción.
4. Duración. Minutos que dura el examen como máximo.
5. * (1) (5) Fecha Tope
6. * (5) Hora Tope.
7. Lista de Preguntas. Cada pregunta constará de:
    i. Descripción.
    ii. Lista de Respuestas Posibles. Cada respuesta constará de:
        1. Descripción.
        2. Correcta – un checkbox para marcar si es correcta esta respuesta o no (puede haber más de una respuesta correcta por pregunta). Solo se considera correcta en caso de que se hayan marcado TODAS las respuestas de la pregunta.
8. *Porcentaje de aprobación. Por ejemplo 70% querrá decir que alcanzando o superando ese porcentaje, el examen habrá sido aprobado.

* Campos Requeridos.
(5) pueden campos distintos en el formulario pero en la Base de Datos debe de guardarse como un solo campo del tipo datetime.

El profesor deberá visualizar una grilla de todos los exámenes de los cursos en los que es profesor. La misma tendrá las siguientes columnas:

1. Curso
2. Examen. Nombre de examen
3. Fecha y Hora Tope
4. Rindieron. Cantidad de alumnos del curso que rindieron.

5. Aprobados. Cantidad de alumnos que aprobaron el examen.
6. Reprobados. Cantidad de alumnos que desaprobaron el examen.
7. Faltan. Cantidad de alumnos del curso que faltan rendir.

5. Lista de exámenes (para Alumno)

El alumno al ingresar deberá tener un listado de los exámenes de los cursos a los que pertenece. La grilla contendrá las siguientes columnas:

1. Curso
2. Examen. Nombre de examen
3. Fecha y Hora Tope
4. Resultado.
    a. En caso de que el alumno no lo haya hecho y que la fecha actual sea mayor que la fecha y hora tope se mostrará “Examen Vencido”.
    b. En caso de que la fecha actual sea menor que la fecha y hora tope se mostrará un Link/Botón “Realizar Examen”
        i. En este caso el alumno es redirigido a otra página - ver Realizar examen (para Alumno)
    c. En caso de que el alumno lo haya hecho, deberá mostrarse “Aprobado”, “Reprobado”.

6. Realizar examen (para Alumno)

El alumno verá un encabezado con el nombre del examen, su descripción (en caso de que tenga), duración (en caso de que tenga), junto con un link/botón de “Volver” que vuelve a 5. Lista de exámenes y un botón de “Iniciar”.

i. Al iniciar se deberá mantener el encabezado y debajo mostrar la primera pregunta del examen, el listado de respuestas posibles, junto con un link/botón de siguiente.
ii. Al presionar siguiente, al alumno se le mostrará la siguiente pregunta y así sucesivamente.
iii. Al presionar siguiente y ya no quedar más preguntas, se lo redirigirá a una página de resultado - ver Resultado de examen (para Alumno).
iv. En caso de haberse superado el tiempo estimado para el examen (duración) o de haberse superado la Fecha y Hora Tope, el examen se dará por realizado y se redirigirá a la página de resultado - ver Resultado de examen (para Alumno).
v. No es requerido que funcione el caso en que un alumno intente hacer 2 exámenes simultáneamente.

Nota: Esta página deberá ser realizada utilizando solo HTML, JS y CSS, sin utilizar controles ASP.Net. Para la carga de contenido e interacción se debe consumir un Web Service con json (ver API de Examen) a través de javascript (por ejemplo utilizando jquery Ajax http://api.jquery.com/jQuery.ajax/). Este javascript deberá recibir la respuesta del web service y armar/cargar el correspondiente html

Sugerencia:
Una opción para simplificar la lógica seria que no se almacenen las respuestas individuales de los alumnos en la Base de Datos, a medida que va respondiendo cada pregunta, se compara las respuestas marcadas con las marcadas como correctas por el profesor y se guarda en una variable en **Session** con la cantidad de respuestas correctas hasta el momento.
Luego al contestar la última pregunta o finalizado el tiempo, se compara la cantidad de respuestas correctas respecto de la cantidad total de preguntas y del porcentaje necesario para la aprobación, y se borra lo guardado en esta variable.
Esta información se debe guardar en una tabla **ResultadoExamen** el resultado del examen para ese alumno, guardando, IdResultadoExamen, IdExamen, fecha y hora de inicio, tiempo demorado, fecha y hora fin, idAlumno, porcentaje obtenido, resultado.

7. Resultado de examen (para Alumno)

El alumno verá el nombre del examen, fecha y hora de inicio del examen, email del alumno, tiempo que le tomo hacer el examen (minutos), fecha y hora de fin del examen, el porcentaje que obtuvo, resultado del mismo (aprobado/reprobado) y un link/botón que diga “Volver a Exámenes Disponibles” que redirige a 5. Lista de exámenes

8. Api de Examen

Se creará una API para todas las operaciones necesarias en la página de Realizar Examen. Todos los métodos deberán de devolver un objeto o lista de objetos que sean de una clase creada por nosotros, no deben ser tipos primitivos como string, bool, int, etc.
Toda respuesta debe tener al menos las propiedades bool OperacionExitosa (indicando si la operación falló o no), string Mensaje (en caso de haber un error, deberá mostrarse aquí el mismo, caso contrario esta vacia), además de lo que se desea devolver como respuesta, para ello se debe utilizar Herencia/Composición/Generics.

Por ejemplo

public Pregunta ObtenerProximaPregunta (idExamen, preguntaAnterior)

Donde Pregunta puede heredar de una clase RespuestaBase o puede tener una propiedad del tipo RespuestaBase o el método podría devolver RespuestaBase<Pregunta> utilizando generics.

REQUISITOS GLOBALES

Requisitos No Funcionales

Compatibilidad con exploradores.

- Internet Explorer 9,10
- Firefox (la última versión para Windows).
- Google Chrome (la última versión para Windows).

Errores

- Ninguna Excepción en momento de ejecución puede ser vista por ningún usuario del sitio.
    - (*) Pagina No Encontrada:
        - Se debe de crear una página para los errores del tipo 404 (página no encontrada).
    - (*) Descripciones de Error:
        - Los usuarios del sitio solo deben de ver mensajes amigables.
        - Se debe de crear una página para cualquier otro error arrojado por la aplicación y que no haya sido atrapado.
    - Logging:
        - Debe de existir un sistema de logging de errores, donde se guarde la fecha, url del error y toda la información relacionada al error para facilitar su corrección. Este logging puede hacerse guardando la información en una tabla o se puede guardar en un archivo (una opción es utilizar la biblioteca Log4Net)

* Esto se debe implementar utilizando la configuración de custom errors en el web.config

ENFOQUE

Diseño y Arquitectura

1. Estilo
    a. Todos los archivos .css deberán estar dentro de una carpeta.
    b. No utilizar estilos inline (atributo style="") ni definir estilos dentro de una página (tags <style>).
    c. Debe de utilizarse algún framework/biblioteca de hojas de estilo. Algunos ejemplos:
        i. Twitter Bootstrap (http://getbootstrap.com/ , temas http://bootswatch.com/)
        ii. Foundation (http://foundation.zurb.com/docs/)
        iii. KickStart (http://www.99lime.com/elements/)
        iv. otro
2. JavaScript
    a. No utilizar JavaScript inline dentro de una página, se deberá referenciar a archivos js.
    b. Todos los archivos .js deberán estar dentro de una carpeta (por ejemplo /scripts o /js).
        i. Si se decide utilizar algún js que no es propio, el mismo deberá estar dentro de una subcarpeta.
        ii. Las funciones específicas de una página, deberían estar en un archivo .js con el mismo nombre de la página.
    c. Aquellas funciones utilizadas en más de una página, deberían de estar dentro de otro archivo .js.
    d. Debe utilizarse JQuery.

3. HTML
    a. No utilizar tags table.
    b. Todo el contenido dentro debe ser SEO Friendly.
        i. Debe existir un archivo sitemap.xml (referencia: http://www.sitemaps.org/protocol.html)
    c. Debe utilizarse Master Pages.
    d. Debe utilizarse xHtml.
4. Validación
    a. Utilizar validaciones tanto del lado del cliente (JavaScript) como del lado del servidor.
    b. Se puede utilizar una lista que detalle todos los campos que no cumplieron con las validaciones.
5. Solución
    a. La solución debe estar separada en capas, al menos 3 (presentación, negocio/servicios y capa de datos). Cada una de estas capas debe estar en un proyecto de biblioteca de clases (o "class library") distinto.

Técnico

1. Código:
    a. Utilizar la menor cantidad posible de código en los archivos aspx.cs, ascx.cs, master.cs, etc. e intentar que en los mismos haya llamadas a métodos dentro de otro proyecto que contenga las reglas de negocio.
    b. Las página **1. Login de Profesor** y **2. ABM de Curso (Solo Profesor)** deben ser ASP.NET puras (con controles ASP.NET).
    c. La página **6. Realizar examen (para Alumno)** debe estar hecha sin controles ASP.Net - ver Nota en 6. Realizar examen (para Alumno)
    d. El resto de las páginas pueden o no estar hechas con controles ASP.Net.
2. Seguridad
    a. Encriptar información sensible, como la contraseñas.

Aquellos grupos que sean de 3 Alumnos, deberán además hacer

1. El Login de Alumnos debe poseer un Captcha random (que es un Captcha?)
2. En **4. Administración de Exámenes (para Profesor)** deberán de agregar una columna que sea un link/botón "Ver Gráficos" que los redirigirá a una pantalla donde se verán 2 gráficos de torta (ejemplo: http://www.jqplot.com/tests/pie-donut-charts.php) para rindieron/faltan y el otro para aprobados/reprobados junto con un link/botón de "Volver".
3. En caso de que un examen se haya iniciado pero el sistema no detectó que llego a terminar nunca (por ejemplo se cerró el navegador), el resultado del mismo deberá de evaluarse como si se hubiese completado, tomando las respuestas correctas hasta el momento y las faltantes como si se hubiesen contestado mal.
    a. Para esto deberían de capturar el evento Session_End en la clase dentro de global.asax.cs y verificar si en Session existe algún examen en curso para el alumno y en ese caso guardar en la Base de Datos el resultado.