IF 4101: Lenguajes para Aplicaciones Comerciales

Prof. MSI. Esteban Sanabria Mora

Proyecto I - 20%

Fecha de defensa: 27 de mayo del 2021, 1:00 a 9:00pm

Fechas de avances: 6, 13 y 20 mayo

Desarrollo de una SPA Transaccional para la Carrera de Informática Empresarial

Objetivo de negocio:

1. Permitirle a la carrera de Informática Empresarial del Recinto de Paraíso contar con información centralizada de estudiantes, profesores, cursos, así como de noticias y temas de interés, mediante una aplicación web a la medida, con el fin de poder realizar consulta y transacciones propias del proceder académico y recreativo de la carrera.

Objetivos técnicos:

1. Utilizar tecnologías actuales de desarrollo web como .Net, C#, Razor, MVC, API Core, Javascript, etc para la implementación de un sitio web transaccional que les permita a los docentes, estudiantes y administradores del sitio realizar sus operaciones y obtener la información que buscan.
2. Implementar herramientas de versionamiento de código y de integración continua que les permita a los desarrolladores trabajar remotamente y centralizar sus desarrollos, mediante la utilización de la nube o de un servidor virtual local con una VPN.
3. Aplicar pruebas **automatizadas** de software que garanticen la calidad del desarrollo y la completitud de los requerimientos, mediante el uso de herramientas como Selenium o similar.

Descripción de las necesidades de negocio:

Imagine que su grupo de trabajo es una empresa de desarrollo de software que fue contratada por el Recinto de Paraíso para construir una aplicación web enriquecida y transaccional para la carrera de Informática Empresarial del Recinto. La idea es que la carrera cuente con un sitio web tanto de consulta como de apoyo a gestiones de su estudiantado y de su cuerpo docente.

La aplicación incluye las siguientes funcionalidades:

1. Registro de estudiantes: Los estudiantes acceden el sitio y se registran. Se envía un correo al estudiante indicando que su solicitud está en espera de aprobación.
2. Aprobación de registros: El administrador del sitio aprueba o rechaza las solicitudes de registro de los estudiantes (se envía correo al estudiante indicando la aprobación o rechazo).
3. Registro de profesores por parte del administrador: Se les envía correo indicando que han sido registrados.
4. Registro de cursos: el administrador registra los cursos de la carrera y determina cuáles están activos y cuáles no, dependiendo del semestre. Solo los cursos activos aparecen en el sitio (o bien, todos aparecen y solo los activos salen habilitados). Debe poderse hacer CRUD de cursos. **Nota:** Indicarle al docente una propuesta de los datos que lleva Curso, incluyendo el profesor.
5. Autenticación en el sitio: para realizar tareas como modificación de perfil, envío de consultas, comentarios, aprobaciones, etc, el estudiante/docente/admin debe estar autenticado en el sistema. La autenticación se hace con base en el username y password con el que se hayan registrado.
6. Perfil de los estudiantes y docentes: Aparte de los datos básicos ingresados en el registro de estos, ellos pueden actualizar sus perfiles con fotografía, formación profesional (en el caso de docentes), link a redes sociales/profesionales, intereses, etc. **Nota:** Indicarle al docente una propuesta de los datos que incluirán en perfil del estudiante y del docente. El estudiante o docente puede eliminar su cuenta y darse de baja en el sistema, sin embargo, todo lo indicado antes por el estudiante/docente en comentarios, etc se mantendrá.
7. Noticias de interés: El administrador puede publicar noticias del acontecer académico y de la carrera. Además, para este apartado en particular, el presidente de la asociación de estudiantes de la carrera tendrá la posibilidad de publicar las

noticias de ámbitos recreacional, derechos de estudiantes, guías de horarios y profesores, etc. Es decir, estas noticias podrían ser texto, pero también pueden incluir archivos y fotografías. Se debe poder hacer CRUD de las noticias.

8. Comentarios sobre noticias: Los estudiantes y docentes podrán comentar las noticias, pero el administrador podrá eliminar los comentarios cuando estos se pasen de tono.
9. Envío público o privada de consultas o de inquietudes: los estudiantes podrán enviar públicamente o de forma privada consultas respecto a cursos. Las que son de forma pública podrán ser visualizadas y comentadas por toda la comunidad y las que no, serán vistas por el docente a cargo del curso y por el admin. Estos podrán darle respuesta.
10. Solicitud de citas de atención: los estudiantes podrán solicitar a los docentes citas para ser atendidos en horas de consulta, tanto virtual como presencialmente. Los docentes serán notificados por correo sobre el ingreso de estas citas y podrán también aceptarlas o rechazarlas indicando algún comentario en ambos escenarios. Los docentes verán, por día de consulta, cuáles estudiantes tienen para atender.

Aspectos por considerar en cuanto a programación:

- Deben utilizar .Net C# como framework/lenguaje base.
- Utilicen a discreción vistas hechas con Razor o elaboradas “a pie”.
- Todo el módulo de noticias se hace con API. El resto con MVC normal.
- Deben utilizar EntityFramework tanto en la API con en la app MVC.
- Los catálogos que NO tengan CRUD (Ejemplo: provincia, cantón, distrito en perfil) deben “popularse” mediante LINQ o mediante la forma directa de llamar a SPs o consultas en BD.
- Deben crear pruebas automatizadas para cada caso de uso. Estas se ejecutan cada vez que nuevas funcionalidades se integren, con el fin de asegurar la calidad de lo desarrollado.

Diseño gráfico y arquitectónico:

- Deben utilizar una plantilla responsiva que sea de tipo SPA y la paleta de colores debe empatar con los colores de la UCR (pueden modificar una). Se les insta a utilizar la que se trabajó en laboratorios anteriores.

- Deben cumplir con las normas de desarrollo web que propone la W3C. Todo código de frente debe primero probarse con el W3C tester antes de ser integrado al proyecto.
- El sitio web debe cumplir con los postulados de la arquitectura de información web: etiquetado, búsqueda, navegabilidad y organización. Consultar al docente si tienen dudas al respecto.
- Deben tener dos bases de datos en MS SQL Server (una para MVC y otra para API)
- Pueden utilizar el servidor de base de datos de la sede, pero deberán tener un backup actualizado por si este falla.
- Pueden utilizar herramientas de versionamiento de su gusto: Team Foundation, Git, etc. Lo mismo con herramientas de continuos integration. Pueden ser en la nube o locales (mientras tengan montada una VPN). NO se revisan ni avances ni proyectos que no estén versionados. Compartir el repo con el docente.

Management de cada equipo:

- **Todos** los estudiantes deben desarrollar. Cada uno debe tener una cuenta en GIT, TFS, etc y en la herramienta de integración continua de su interés y realizar periódicamente sus aportes de código. Sin embargo, les solicito que cada miembro tenga un rol preponderante: Front-end developer, back-end developer, QA automation engineer, etc. Traten de que solo una persona manipule las BDs. Ese rol es el del DBA.
- **Cada equipo debe nombrar a un PM.** Esta será la persona con que el docente, en modo cliente, tendrá comunicación y quien se le solicitará que muestre avances del proyecto (todo el resto del equipo).
- Deben crear un cronograma de actividades y de distribución de tareas. Utilicen Trello o alguna otra herramienta de gestión que les permita controlar el avance de su proyecto. Descarguen la herramienta Slack para comunicación con su equipo. Es bastante útil y así separan un poco sus WhatsApp de la academia.
- Cada semana se revisa avance basado en lo que se indicó que se completaría, en el cronograma.
- Los equipos de trabajo pueden ser parejas, tríos, o cuartetos. No más que eso. Se puede trabajar individualmente **bajo su propio riesgo**.

Componentes de evaluación:

- Avances de proyecto semanales y consultas al docente ……..……..15%

- Utilización de herramientas de integración continua y versionamiento.....15%
- Implantación correcta y completa de los requerimientos ……..……..…….50%
- Management (cumplimiento de cronogramas, roles, etc)……10%
- Defensa del proyecto frente al cliente…….....10%