PROYECTO DE CARNET DIGITAL

Programación V

Colegio Universitario de Cartago

Prof. Ing. Gabriel González Solano

Carnet Digital del CUC

Objetivo general proyecto

Realizar la experiencia de la elaboración de un proyecto que se espera salga a producción, pasando por todas las etapas del desarrollo de software.

Objetivos específicos

- Analizar requisitos de desarrollo de distintos componentes de software.
- Diseñar la propuesta de solución para los distintos componentes de software.
- Realizar la programación de los componentes de software que formarán parte de la solución.
- Probar los distintos componentes de software que formarán parte de la solución.
- Implementar los componentes de software que formarán parte de la solución.
- Trabajar en equipo para alcanzar la consecución de las metas del proyecto.
- Utilizar software que facilite la integración de código fuente y la gestión del ciclo de vida del software.

Contexto del proyecto

Recientemente y por directriz de la decanatura, el CUC ha fortalecido las medidas de seguridad en el ingreso a la institución, por medio del uso del carnet institucional, tanto para funcionarios como para estudiantes. Como toda medida de seguridad que se aplica, conlleva la aparición de algunos inconvenientes para las personas en las que se aplica esta.

El principal inconveniente que se ha presentado es cuando una persona olvida su carnet físico pues debe realizar un trámite que retrasa su ingreso a la institución y que provoca que el personal de seguridad deba atender la situación.

Como parte de las iniciativas que surgen a lo interno de nuestra institución, se ha planteado la posibilidad de realizar un proyecto que cubra los temas del curso de programación V y que permita al final de su ejecución poseer la mayor parte de la solución del Carnet Digital del CUC, el cual se espera que permita a las personas portar su identificación institucional en un dispositivo móvil y al personal de seguridad poder corroborarlo de una manera sencilla.

Arquitectura general del sistema

A continuación, se presenta un vistazo arquitectónico de alto nivel de cada uno de los componentes que formarán parte de la solución a implementar:

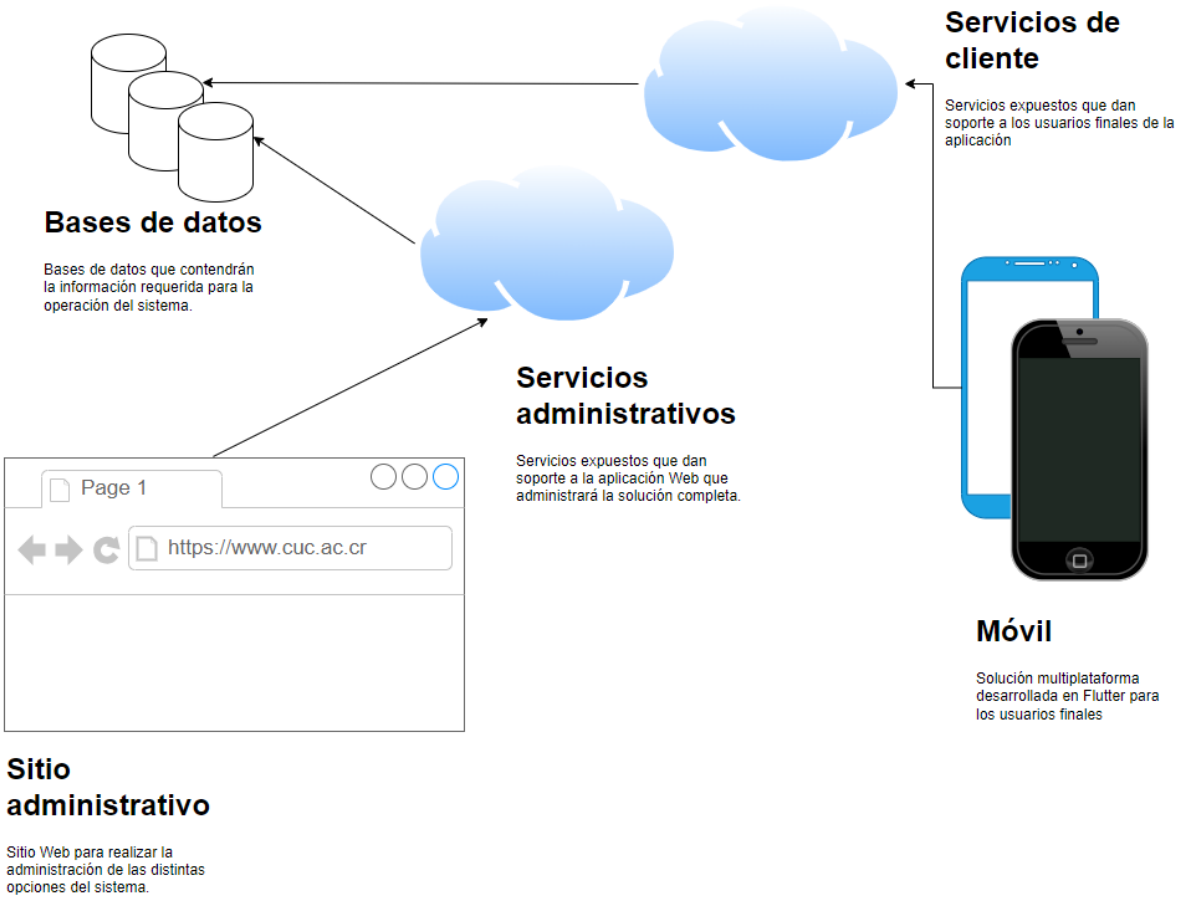


Ilustración 1 Arquitectura general del sistema

Organización del trabajo y de los equipos

Para trabajar el proyecto se seguirá el modelo de trabajo propuesto en Scrum. Dada la importante cantidad de requerimientos a atender, cada uno de los subgrupos formados realizará la atención de diferentes componentes.

El grupo completo se subdividirá en 3 grupos que lo conoceremos como unidad. Cada unidad se asignará un nombre de grupo por medio del cual será reconocido y lo indicará el primer día de clases. Cada grupo de trabajo se distribuirá las historias de usuario que se le asignen entre sus integrantes. Si bien es cierto al final a responsabilidad es individual, cada grupo es responsable de las historias que se le asignen para lograr un sprint exitoso.

Historias de usuario

Para la atención de los requisitos a desarrollar para el sistema, se crearán una serie de historias de usuario donde se detallará qué espera el usuario del sistema. A continuación, se realiza el detalle de las historias de usuario y sus criterios de aceptación.

Primer alcance - Servicios Rest

ID	Historia de usuario	Criterios de aceptación
SRV1	Yo como arquitecto del sistema quiero microservicios para administración de usuarios.	• Deben crearse 6 operaciones que permitan la administración de usuarios y que cumplan con el estándar Rest. Estas operaciones deben responder en el endpoint /usuario y deben permitir: ○ Crear un usuario. ○ Modificar un usuario. ○ Eliminar un usuario. ○ Obtener los datos de todos los usuarios. ○ Obtener los datos de un usuario por su llave primaria. ○ Obtener datos de los usuarios filtrados por: identificación, nombre, tipo.

		• Los datos que se requieren para un usuario son:
		  ○ Email (será la identificación única del usuario).
		  ○ Tipo de identificación. Debe permitirse que eventualmente existan más tipos de identificación.
		  ○ Identificación.
		  ○ Nombre completo.
		  ○ Contraseña (debe almacenarse encriptada).
		  ○ Tipo de usuario (funcionario, estudiante o administrador). Debe permitirse que eventualmente existan más tipos de usuario.
		  ○ Carreras asociadas (puede ser más de una).
		  ○ Áreas asociadas (puede ser más de una).
		  ○ Teléfono(s) de contacto.
		• Los teléfonos no son obligatorios.
		• Si el usuario es estudiante, tendrá carreras asociadas. Si el usuario es funcionario tendrá áreas asociadas.
		• Debe validarse el formato del email.
		• Los emails deben ser de los dominios cuc.cr o cuc.ac.cr.
		• Si el email es del dominio cuc.cr, el tipo de usuario debe ser estudiante.
		• Si el dominio es cuc.ac.cr, el tipo de usuario debe ser funcionario o administrador.

		• El nombre completo no puede ser vacío ni espacios en blanco. • Todas las operaciones requieren de un token de autorización de usuario autenticado para poder ejecutar, obtenido con el servicio descrito en la HU SRV2. Se debe validar contra el método “validate”.
SRV2	Yo como arquitecto del sistema quiero microservicios de login para autenticar los usuarios que ingresarán al sistema.	• Se requiere de realizar un servicio rest que permita realizar las distintas operaciones de login del usuario, así como de la administración del token para realizar las operaciones. • En el endpoint /login se permitirá que el usuario pueda iniciar sesión, será una operación POST y recibirá en los headers el usuario, la contraseña y el tipo de usuario. Debe verificar si coinciden el usuario, su contraseña y el tipo indicado. Tendrá el siguiente comportamiento: o Si la operación es exitosa el servicio debe responder un 201, junto con la siguiente estructura que permitirá realizar el resto de las operaciones. Este token debe tener una validez de 5 minutos por defecto, pero este periodo debe ser parametrizable. { expires_in = horaVencimiento, access_token = jwtToken, refresh_token = refreshToken, usuarioID = identificador

		}
		○ Si la operación falla el servicio debe responder un 401, junto con un mensaje que indique: “Usuario y/o contraseña incorrectos”.
		○ Todos los datos son requeridos y no pueden ser nulos o blancos.
		○ El refresh token, será un token temporal que permitirá extender el plazo de la sesión del usuario generando un nuevo token para las operaciones. Este token también tiene un periodo de expiración configurable, superior al token JWT.
		• El endpoint /refresh permite obtener un nuevo token. Recibe como parámetro un refresh token y retorna los datos que se indican a continuación en formato json:
		{
		expires_in = horaVencimiento,
		access_token = jwtToken,
		refresh_token = refreshToken,
		}
		• Si el refresh token es válido se responderá con estado 201 y la estructura anterior. Si no es válido se responderá con un estado 401 con un mensaje que indique: “No autorizado”.
		• El endpoint /validate, recibe como parámetro un token y responderá con el código de estado 200 y un true si es

		válido. De lo contrario responderá con un 401 (Unauthorized).

SRV3	Yo como arquitecto del sistema quiero microservicios para la administración de carreras de la institución	• Deben crearse 5 operaciones que permitan la administración de carreras y que cumplan con el estándar Rest. Estas operaciones deben responder en el endpoint /carreras y deben permitir:
		  ○ Crear una carrera.
		  ○ Modificar una carrera.
		  ○ Eliminar una carrera.
		  ○ Obtener los datos de todas las carreras
		  ○ Obtener los datos de una carrera por su llave primaria.
		• Los datos requeridos para una carrera son:
		  ○ Identificador de la carrera.
		  ○ Nombre de la carrera.
		  ○ Director de la carrera.
		  ○ Email.
		  ○ Teléfono.
		• Todos los datos son requeridos y no pueden ser vacíos ni espacios en blanco.
		• Debe validarse el formato del email.
		• El campo teléfono solo permite valores numéricos.
		• Todas las operaciones requieren de un token de autorización de usuario autenticado para poder ejecutar, obtenido con el servicio descrito en la HU SRV2. Se debe validar contra el método “validate”.

SRV4	Yo como arquitecto del sistema quiero microservicios para la administración de tipos de usuario.	• Deben crearse 5 operaciones que permitan la administración de tipos de usuario y que cumplan con el estándar Rest. Estas operaciones deben responder en el endpoint /tiposusuario y deben permitir:
		  ○ Crear un tipo de usuario.
		  ○ Modificar un tipo de usuario.
		  ○ Eliminar tipos de usuario.
		  ○ Obtener los datos de todos los tipos de usuario.
		  ○ Obtener los datos de los tipos de usuario por su llave primaria.
		• Los datos requeridos para los tipos de usuario son:
		  ○ Identificador del tipo de usuario.
		  ○ Nombre del tipo de usuario.
		• Todos los datos son requeridos y no pueden ser vacíos ni espacios en blanco.
		• Todas las operaciones requieren de un token de autorización de usuario autenticado para poder ejecutar, obtenido con el servicio descrito en la HU SRV2. Se debe validar contra el método “validate”.

SRV5	Yo como arquitecto del sistema quiero microservicios para la administración de tipos de identificación.	• Deben crearse 5 operaciones que permitan la administración de tipos de identificación y que cumplan con el estándar Rest. Estas operaciones deben responder en el endpoint /tiposidentificacion y deben permitir:
		  ○ Crear un tipo de identificación.
		  ○ Modificar un tipo de identificación.
		  ○ Eliminar tipos de identificación.

		○ Obtener los datos de todos los tipos de identificación. ○ Obtener los datos de los tipos de identificación por su llave primaria. • Los datos requeridos para los tipos de identificación son: ○ Identificador del tipo de identificación. ○ Nombre del tipo de identificación. • Todos los datos son requeridos y no pueden ser vacíos ni espacios en blanco. • Todas las operaciones requieren de un token de autorización de usuario autenticado para poder ejecutar, obtenido con el servicio descrito en la HU SRV2. Se debe validar contra el método “validate”.
SRV6	Yo como arquitecto del sistema quiero microservicios para la administración de la fotografía del usuario.	• Deben crearse las siguientes operaciones para la administración de la fotografía: ○ Actualizar fotografía: le agrega o actualiza la fotografía al usuario, la recibe en formato Base 64. Idealmente las imágenes a utilizar no deben ser superior a 1 MB de tamaño y deben estar en formato 4:3. Recibe como parámetros el identificador del usuario y la fotografía. ○ Eliminar fotografía: elimina el dato de la fotografía del usuario. Recibe como parámetro el identificador del usuario. ○ Obtener fotografía: obtiene la fotografía almacenada para el usuario

		en formato Base 64. Recibe como parámetro el identificador del usuario. • Todas las operaciones se deben realizar en el endpoint /usuario/fotografía. • Todas las operaciones requieren de un token de autorización de usuario autenticado para poder ejecutar, obtenido con el servicio descrito en la HU SRV2. Se debe validar contra el método “validate”.

SRV7	Yo como arquitecto del sistema quiero microservicios para la generación de un código QR con la información del carnet.	• Debe crear una operación que permita obtener un código QR con la información del usuario indicado como parámetro.
		• La operación debe responder en el endpoint /usuario/qr.
		• Recibe como parámetro la identificación del usuario.
		• El código QR debe responderse en base 64 y debe incluir en formato json los siguientes datos:
		  ○ Nombre completo.
		  ○ Identificación.
		  ○ Tipo de usuario (descripción del tipo).
		  ○ Carreras o áreas asociadas.
		  ○ Fecha de vencimiento.
		• Todas las operaciones requieren de un token de autorización de usuario autenticado para poder ejecutar, obtenido con el servicio descrito en la HU SRV2. Se debe validar contra el método “validate”.

SRV8	Yo como arquitecto del	• Deben crearse 5 operaciones que permitan la administración de las áreas de trabajo y

	sistema quiero microservicios para la administración de áreas de trabajo de funcionarios.	que cumplan con el estándar Rest. Estas operaciones deben responder en el endpoint /areas y deben permitir: ○ Crear un área de trabajo. ○ Modificar un área de trabajo. ○ Eliminar área de trabajo. ○ Obtener los datos de las áreas de trabajo. ○ Obtener los datos de las áreas de trabajo por su llave primaria. • Los datos requeridos para las áreas de trabajo son: ○ Identificador del área de trabajo. ○ Nombre del área de trabajo. • Todos los datos son requeridos y no pueden ser vacíos ni espacios en blanco. • Todas las operaciones requieren de un token de autorización de usuario autenticado para poder ejecutar, obtenido con el servicio descrito en la HU SRV2. Se debe validar contra el método “validate”.
SRV9	Yo como arquitecto del sistema quiero microservicios que me permita cambiar el estado de un usuario	• Debe crearse una operación que permita modificar el estado de un usuario (activo/inactivo), debe responder por el verbo Patch en el endpoint /usuarios/estado. • Los estados básicos son activo e inactivo, pero debe permitirse que se puedan agregar nuevos estados. • Para cambiar el estado se requiere del identificador del usuario y el código del estado que se desea asignar.

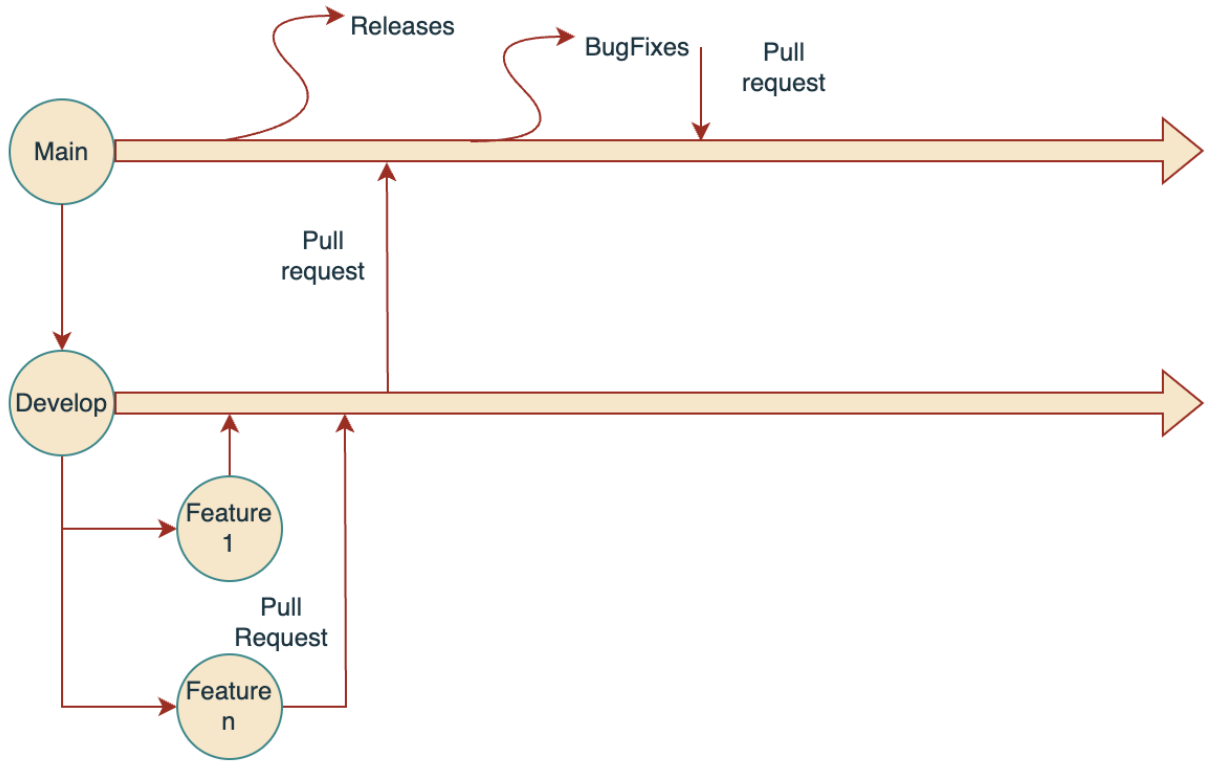
		• Todos los datos son requeridos y no pueden ser vacíos ni espacios en blanco. • Todas las operaciones requieren de un token de autorización de usuario autenticado para poder ejecutar, obtenido con el servicio descrito en la HU SRV2. Se debe validar contra el método “validate”.

Administración de las Historias de Usuario

Las historias de usuario se cargarán en proyectos de alguna herramienta de administración de historias de usuario y seguimiento como Azure DevOps o Gitlab.

Control del código fuente

El código fuente se administrará en una herramienta de repositorio de control de código fuente. En este caso se propone usar repositorio GIT en la herramienta que se acuerde, durante el desarrollo del curso se valorará la mejor herramienta. Se tendrá la siguiente estructura de ramas:



Creación y administración de cambios de la base de datos

La base de datos se tomará el tiempo de definirla en clases, para esto en la segunda lección los distintos equipos formados deberán traer todo el problema analizado y presentar su propuesta de la base de datos. Las mejores ideas de solución se tomarán para la base de datos que se usará en el proyecto.

Si posterior a la definición se necesita de un cambio justificado, debe gestionarse en el grupo de teams del curso, de ser aprobado el que propone creará un script para el cambio y solicitará al profesor aplicar el cambio. Todos los equipos que se vea afectados por este cambio deben ajustar su código fuente para que maneje apropiadamente el cambio.

Tecnologías por utilizar

Para el primer alcance los servicios se desarrollarán en tecnologías .NET 8.0 o bien Node.js y la base de datos a elección del equipo.

Entregables del proyecto

1. Documentación de análisis y diseño (enfocada en las HU que debe atender el equipo).
    a. Portada.
    b. Introducción (Resumen del problema a resolver).
    c. Diagrama de base de datos (completo).
    d. Diagramas de casos de uso.
    e. Diagramas de clases.
    f. Pruebas técnicas.
    g. Conclusiones y recomendaciones.
    h. Bibliografía.
2. Script de base de datos.
3. Código fuente de los servicios desarrollados por el equipo (tomado de la rama de la HU atendida por el equipo de trabajo).
4. Código fuente de los servicios Rest completos (tomado de la rama integrada Develop con los cambios de todos los equipos de trabajo).

5. Historias de usuario actualizadas.
6. Pull Request hacia la rama Main (o rama en GIT donde se encuentra la versión final de su código).

Con respecto a la documentación de pruebas técnicas, en esta actividad debe ejecutar pruebas sobre el servicio de tal forma que se cumpla con las funcionalidades y criterios de aceptación establecidos. Cada criterio de aceptación debe generar una prueba y debe documentarse de la siguiente manera:

# HU	Criterio de aceptación	Evidencia
ID de la HU respectiva	Descripción del criterio que se probó	Pantallazo con la ejecución exitosa.

Aspectos administrativos

1. El proyecto se elaborará en los equipos de trabajo definidos. La calificación será individual.
2. La fecha de entrega del primer alcance será el jueves 12 de junio de 2025.
3. Valor del proyecto 15%.