Tecnólogo Informático

San José de Mayo
14/12/2020

Documento Final

Taller de Sistemas de Información .NET

Integrantes del equipo

Juan Álvarez
juan.alvarez@utec.edu.uy

Lucas Garrido
lucas.garrido@utec.edu.uy

Sebastián Morales
ricardo.morales@utec.edu.uy

Índice

Resumen

En este artículo se presenta el diseño e implementación de una plataforma informática que permita la gestión de una empresa de ómnibus. Para la misma se utilizarán tecnologías del framework .NET y se deberá desplegar en Azure.

Como principal resultado se logró completar con éxito la mayoría de los requerimientos solicitados y se adquirieron conocimientos invaluables.

Palabras clave

.NET Framework, MVC, WCF, WPF, API REST, SOAP

Introducción

El objetivo del laboratorio es que los estudiantes se familiaricen con las distintas tecnologías del stack .NET mediante el diseño e implementación de un sistema de información moderno con una arquitectura distribuida utilizando el abanico de tecnologías .NET.

El sistema a desarrollar consta de un sitio web que permitirá gestionar la información de los empleados de la empresa, de las líneas de transporte y permite a sus usuarios realizar reservas pagando con medios electrónicos. Adicionalmente se desarrollará una terminal de autogestión la cual permitirá a cualquier persona realizar reservas.

El trabajo se plantea en el marco de la edición 2020 de la asignatura Taller de Sistemas de Información .NET.

Marco conceptual

- **API:** (Application Programming Interface) Es un conjunto de subrutinas, funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.
- **CSS:** (Cascading Style Sheets) es un lenguaje utilizado para estilizar elementos escritos en un lenguaje de marcado como HTML. CSS separa el contenido de la representación visual del sitio.
- **Framework:** Es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, que puede servir de base para la organización y desarrollo de software.
- **HTML:** (HyperText Markup Language) Lenguaje de marcado que se utiliza para definir las estructuras de una página de Internet.
- **JavaScript:** Es un lenguaje de programación utilizado principalmente para aportar dinamismo a tiempo real en páginas web.
- **REST:** (Representational State Transfer) es un estilo de arquitectura para diseñar aplicaciones en red, donde las peticiones recibidas son del protocolo HTTP
- **SOAP:** (Simple Object Access Protocol) es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML.
- **SignalR:** Es una biblioteca de software gratuita y de código abierto para Microsoft ASP.NET que permite que el código del servidor envíe notificaciones asincrónicas a las aplicaciones web del lado del cliente.
- **MVC:** Es un framework de aplicaciones web que implementa el patrón modelo-vista-controlador. Basado en ASP.NET, permite a los desarrolladores de software construir una aplicación web como una composición de tres funciones: modelo, vista y controlador.
- **.NET Framework:** Es un framework de Microsoft que hace un énfasis en la transparencia de redes, con independencia de la plataforma de hardware y que permite un rápido desarrollo de aplicaciones.

Descripción del problema

El problema planteado consiste en el diseño y construcción de una aplicación para la gestión de una nueva empresa de transporte público interdepartamental en Uruguay llamada “Uruguay Bus”, la cual quiere tener una fuerte base tecnológica que le permita distinguirse del resto. Alguna de las características que considera fundamental para su puesta en marcha son:

- Brindar una forma ágil de compra de pasajes online.
- Brindar un sistema de visualización online de los vehículos que permita saber a los clientes cuánto tiempo falta para el próximo transporte pase por su parada actual.
- Un fácil control por parte del conductor de los pasajeros que suben, mediante la validación de los pasajes por lectura de código QR.

El sistema requerido estará compuesto por varias componentes de software que serán detalladas a continuación. En el caso de las componentes que tienen una interfaz de usuario se especificarán los casos de uso.

A continuación se listan las funcionalidades agrupadas por rol en el sistema:

- **Invitado**
    - Registro de usuario
    - Iniciar sesión
    - Compra de pasajes (term. autogestión)

- **Usuario**
    - Compra de pasajes (web)
    - Próximos vehículos por una parada
    - Notificaciones de proximidad

- **Conductor**
    - Comienzo de Viaje
    - Finalizar el Viaje
    - Verificación de pasaje

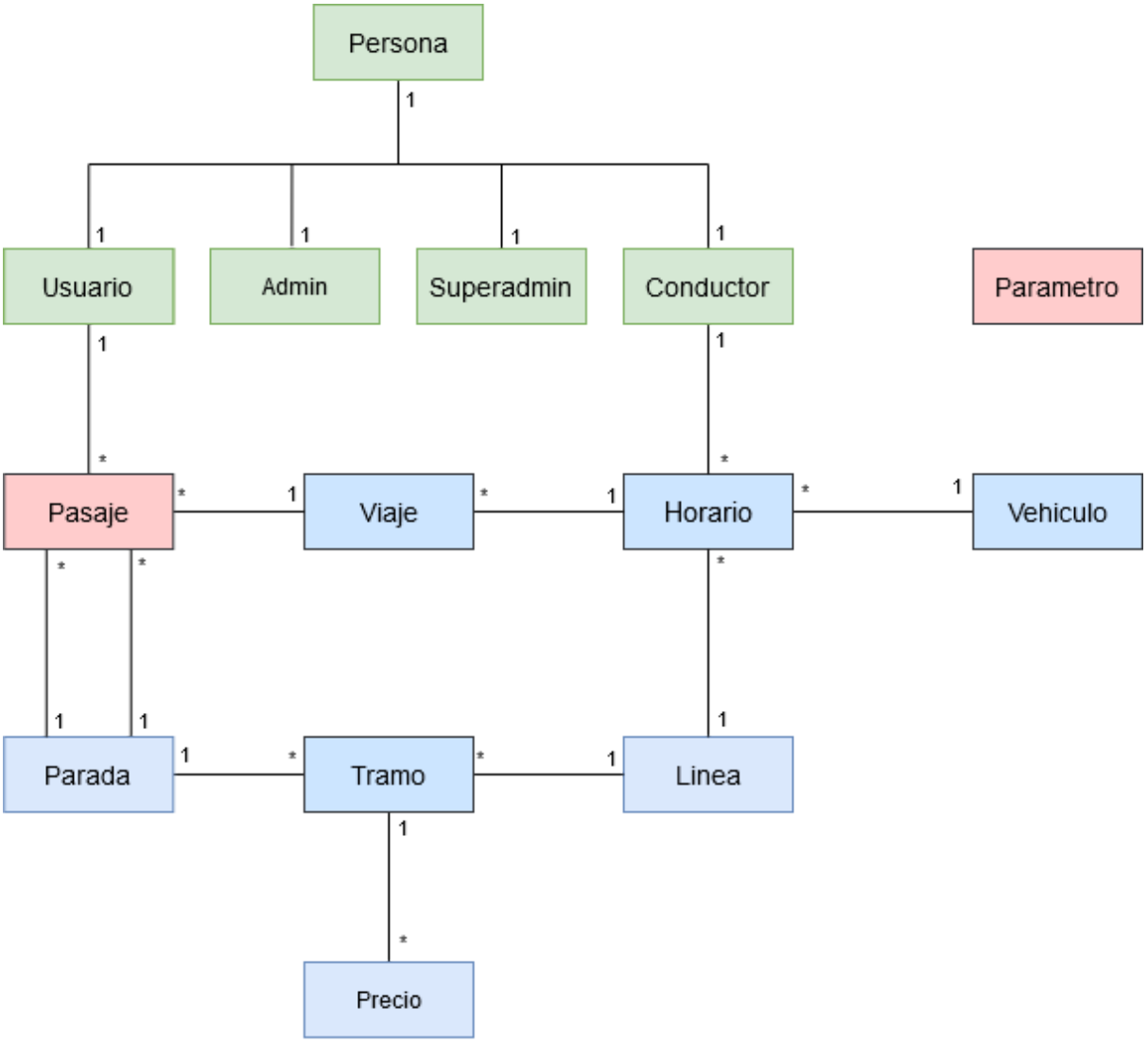
- **Superadmin**
    - Panel de control de vehículos
    - Asignación de Roles

- **Admin**
    - Gestión de vehículos
    - Gestión de paradas
    - Gestión de líneas de transporte
    - Gestión de conductores
    - Gestión de Horarios
    - Gestión de Viajes
    - Reporte de pasajes vendidos
    - Reporte de porcentajes de utilidad

Solución planteada

La funcionalidad central del sistema está dada por la venta online de pasajes para los usuarios registrados. Los usuarios podrán buscar el viaje que más le convenga según la línea y horario que se adapte a su necesidad.

El sistema cuenta con múltiples roles, el principal es el rol “USUARIO”, el cual se asigna automáticamente al momento de registrarse en el sitio web. El usuario con rol “ADMIN” es quien gestiona la aplicación y tiene los permisos necesarios para registrar la información de los servicios que brinda la empresa. El rol “CONDUCTOR” se le asigna a los conductores de los vehículos que realizan los viajes. Por último el rol “SUPERADMIN” es el que posee los mayores privilegios del sistema.



Modelado de la realidad

En lo referido a los requerimientos funcionales, se logró la implementación de la amplia mayoría. Las funcionalidad que no lograron ser cubiertas son las relacionadas a la generación de reportes y a las notificaciones de proximidad.

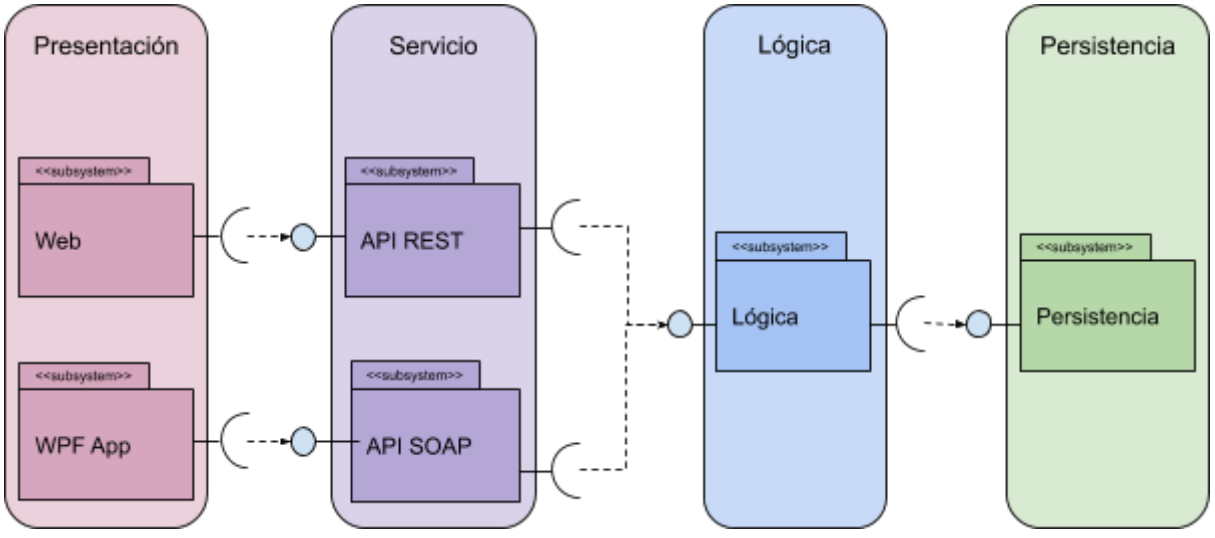
Arquitectura del sistema

El sistema desarrollado posee una arquitectura en capas donde cada capa consume recursos de la capa inferior.

La capa de persistencia se encargará del almacenamiento de la información mientras que en la capa lógica se realizan los controles de integridad referentes a la lógica de negocio.

En la capa de servicio se encuentran dos módulos que expondrán funcionalidades a la capa de presentación. Un módulo consta de una API REST la cual expone todos los recursos necesarios para el sitio web, y una API SOAP la cual expone funciones específicas para la reserva de pasajes en las terminales de autogestión.

En la capa de presentación se dispone del módulo web el cual permitirá a los usuarios registrados realizar acciones según su rol en el sistema. En esta capa también se encuentra la aplicación utilizada en las terminales de autogestión para la reserva de pasajes.



Implementación

Productor y herramientas

Como principal editor de código se utilizó Microsoft Visual Studio, Git y GitKraken como gestores de repositorios y SQL Server como motor de base de datos

Herramienta o producto	Puntos Fuertes	Puntos Débiles	Evaluación General
Microsoft Visual Studio	Buen Autocompletado, gran cantidad de complementos y un gran depuración	Alto consumo de recurso	Una gran IDE para desarrollar de manera eficiente
MVC	Múltiples vistas a partir del mismo modelo	Tiene una gran curva de aprendizaje.	Se esta quedando atras con las nuevas tecnologías
WPF	Una gran cantidad de componentes para el desarrollo	No es fácil tener configuraciones para release y para debug por separado	Una manera de crear aplicaciones muy rapido y facil
WCF	Facilidad de crear y de integrar a proyectos	Tecnología propietaria	Una manera fácil y rápida de separar la lógica de negocios
GitKraken	Facilita la comprensión del estado del proyecto y su gestión	No es gratuito para repositorios privados	Altamente recomendable, muy buena herramienta
SQL Server	Es útil para manejar y obtener datos de Azure y fácil de integrar con sistemas desarrollados en .net	Es difícil limpiar la base de datos después de hacer pruebas	Buena herramienta a la hora de desarrollar con .net
Azure	Fácil a la hora de publicar el proyecto en la plataforma	No brinda ninguna opción de prueba gratuita	Buena herramienta para hostear cualquier sitio

Problemas encontrados

El principal problema que tuvimos fue la falta de tiempo para la investigación de las tecnologías y el desarrollo del proyecto. Esto llevó a la necesidad de realizar una planificación con el fin de disminuir las repercusiones sobre el proyecto. De todas formas este problema atrasó todo lo planificado en un comienzo por lo que no logramos completar el 100% de las funcionalidades

Evaluación de la solución

En lo que a mantenibilidad y escalabilidad concierne, el diseño realizado permite la fácil adición de nuevas y funcionalidades, pero dado que el componente de páginas web es independiente del componente de la API REST, al momento de implementar nuevas funcionalidades, se deben actualizar estos componentes por separado. Esto puede ser visto como una ventaja dado que se logra cierta independencia entre los componentes mencionados.

Al haber dividido el proyecto en diferentes componentes, es fácil de desplegar de manera distribuida, especialmente si se utiliza los servicios de Microsoft Azure.

Como punto negativo se puede mencionar que no se ha utilizado una tecnología SPA lo cual hace que la navegación entre páginas sea poco eficiente y amigable. Adicionalmente se puede agregar que el diseño del sitio se basa sobre una plantilla predefinida y no es un diseño propio.

Desarrollo del proyecto

El desarrollo del proyecto fue ágil y constante, a pesar de los contratiempos mencionados en secciones anteriores.

La cantidad de horas previstas y dedicadas para la realización de las actividades son los siguientes: (horas por persona)

	Horas previstas	Horas dedicadas
Investigación	35	55
Documentación	10	15
Análisis y diseño	10	15
Implementación	55	103
Testing	10	12
Total	**120**	**200**

Las horas dedicadas tuvieron una desviación aproximada de 60% por sobre lo planificado inicialmente, lo cual fue producto de la necesidad de incrementar el tiempo dedicado en las diferentes etapas del proyecto.

Conclusiones y trabajo a futuro.

En conclusión se cumplió con la entrega logrando implementar las funcionalidades fundamentales de la plataforma desarrollada. Se aplicaron los conocimientos aprendidos en clase más los conocimientos adquiridos durante la investigación del stack de tecnologías .NET Framework.

Como trabajo a futuro se puede mencionar la implementación de las funcionalidades que no pudieron ser abordadas para esta entrega. Otro aspecto a mejorar es el diseño tanto de las páginas del sitio web como de la terminal de autogestión.

Referencias

1. .NET Framework - [consultado: Diciembre 2020]
   https://docs.microsoft.com/en-us/dotnet/
2. ASP .NET MVC - [consultado: Diciembre 2020]
   https://docs.microsoft.com/en-us/aspnet/mvc/
3. SignalR - [consultado: Diciembre 2020]
   https://docs.microsoft.com/en-us/aspnet/signalr/overview/getting-started/introduction-to-signalr
4. MercadoPago - [consultado: Diciembre 2020]
   https://www.mercadopago.com.uy/developers/en/guides/online-payments/checkout-api/receiving-payment-by-card/
5. Azure - [consultado: Diciembre 2020]
   https://docs.microsoft.com/en-us/azure/?product=featured