UNIVERSIDAD DE LAS AMÉRICAS

FACULTAD DE INGENIERÍA

CARRERA DE INGENIERÍA EN SOFTWARE

Integrantes: Sebastian Abad, Daniel Cornejo y Joseph Flores

Fecha de entrega: 27/10/2024

EASY TICKET

Propuesta.-

Este proyecto desarrollará una aplicación web basada en el modelo MVC, cuyo objetivo es proporcionar una plataforma donde los usuarios puedan revender entradas para eventos deportivos o conciertos en caso de no poder asistir. La aplicación permitirá a las personas que no lograron adquirir boletos a tiempo o que decidieron asistir en último momento acceder a una opción confiable de compra. Con esta herramienta, se busca evitar la pérdida de dinero por inasistencia a eventos y brindar una alternativa segura y conveniente para la reventa de entradas.

Requerimientos Funcionales.-

Requerimiento 1:

El sistema debe mostrar una página de inicio con un menú principal que incluya:

Una breve descripción de la plataforma.

Un botón de "Home" que permita regresar a la página principal desde cualquier sección.

Un botón de "Tickets" que redirija a la lista de todos los tickets.

Un apartado en la página principal que muestre las tres entradas más recientes registradas en la plataforma.

Requerimiento 2:

Al hacer clic en el botón "Tickets", el sistema debe mostrar una lista completa de todos los tickets disponibles. Esta lista debe ordenarse de forma descendente, mostrando primero los tickets más recientes y, en orden, los tickets menos recientes.

Requerimiento 3:

El sistema debe permitir que el propietario de un ticket edite los detalles del mismo. Para esto, el propietario debe:

Seleccionar el ticket deseado en la lista.

Ingresar una contraseña para autenticarse como dueño del ticket.

En caso de que la contraseña sea incorrecta, el sistema debe:

Mostrar un mensaje de error indicando que la contraseña es incorrecta.

Permitir al usuario volver a intentar la autenticación.

Si el propietario abandona el proceso de autenticación o no completa la autenticación en un tiempo determinado:

El sistema debe cancelar el proceso de edición.

Regresar al usuario a la lista de tickets.

Si la contraseña es correcta, el sistema debe permitir al propietario modificar los detalles del ticket.

Requerimiento 4:

El sistema debe permitir que el propietario de un ticket elimine su publicación. Para ello, el propietario debe:

Seleccionar el ticket deseado en la lista.

Ingresar una contraseña para autenticarse como dueño del ticket.

En caso de que la contraseña sea incorrecta, el sistema debe:

Mostrar un mensaje de error indicando que la contraseña es incorrecta.

Permitir al usuario volver a intentar la autenticación.

Si el propietario abandona el proceso de autenticación o no completa la autenticación en un tiempo determinado:

El sistema debe cancelar el proceso de eliminación.

Regresar al usuario a la lista de tickets.

Si la contraseña es correcta, el sistema debe permitir la eliminación del ticket.

Casos de Uso.-

Caso de Uso 1: Visualizar Menú Principal

Descripción: El sistema muestra la página principal con la descripción, botones de navegación y una sección con los tres tickets más recientes.

Actores: Usuario

Flujo Principal:

El usuario accede a la aplicación.

El sistema muestra la página principal con:

Descripción de la plataforma.

Botón de "Home".

Botón de "Tickets".

Las tres entradas más recientes.

Precondiciones: El sistema está disponible y accesible.

Postcondiciones: La página principal es visible para el usuario.

Caso de Uso 2: Visualizar Todos los Tickets

Descripción: El usuario puede acceder a una lista de todos los tickets disponibles, ordenados de más recientes a menos recientes.

Actores: Usuario

Flujo Principal:

El usuario selecciona el botón "Tickets" en el menú principal.

El sistema muestra la lista de tickets ordenada de más recientes a menos recientes.

Precondiciones: Existen tickets registrados en el sistema.

Postcondiciones: El usuario puede ver todos los tickets en el sistema, ordenados de manera adecuada.

Caso de Uso 3: Editar un Ticket

Descripción: El propietario de un ticket puede editar sus detalles, previa autenticación con una contraseña.

Actores: Usuario propietario autenticado

Flujo Principal:

El propietario selecciona el ticket que desea editar.

El sistema solicita la contraseña para confirmar la propiedad.

El propietario ingresa la contraseña.

El sistema valida la contraseña.

Si la contraseña es correcta, el sistema permite al propietario editar los detalles del ticket.

El propietario realiza las modificaciones y guarda los cambios.

El sistema guarda las modificaciones y muestra el ticket actualizado.

Flujos Alternativos:

Flujo Alternativo 3.1: Contraseña incorrecta.

El sistema muestra un mensaje de error indicando que la contraseña es incorrecta y permite al usuario volver a intentarlo.

Flujo Alternativo 3.2: El propietario abandona el proceso de edición presionando en "Ticket".

El sistema cancela la operación de edición y regresa al usuario a la lista de tickets.

Precondiciones: El usuario es el propietario del ticket y conoce la contraseña.

Postcondiciones: El ticket se actualiza con los cambios realizados, o la operación se cancela.

Caso de Uso 4: Eliminar un Ticket (Propietario)

Descripción: El propietario de un ticket puede eliminar su publicación, previa autenticación con una contraseña.

Actores: Usuario propietario autenticado

Flujo Principal:

El propietario selecciona el ticket que desea eliminar.

El sistema solicita la contraseña para confirmar la propiedad.

El propietario ingresa la contraseña.

El sistema valida la contraseña.

Si la contraseña es correcta, el sistema permite al propietario eliminar el ticket.

El sistema elimina el ticket y actualiza la lista de tickets.

Flujos Alternativos:

Flujo Alternativo 4.1: Contraseña incorrecta.

El sistema muestra un mensaje de error indicando que la contraseña es incorrecta y permite al usuario volver a intentarlo.

Flujo Alternativo 4.2: El propietario abandona el proceso de eliminación.

El sistema cancela la operación de eliminación y regresa al usuario a la lista de tickets.

Precondiciones: El usuario es el propietario del ticket y conoce la contraseña.

Postcondiciones: El ticket se elimina del sistema, o la operación se cancela.

ENFOQUES.-

Enfoque 1: Code-First

En el enfoque Code-First, el desarrollo de la aplicación comienza por definir las clases del modelo en código, lo que nos permite como los desarrolladores enfocarse primero en el diseño lógico. A partir de estas clases, se genera la base de datos en el sistema mediante migraciones.

Ventajas: Este enfoque es útil cuando se parte de cero o el modelo cambia con frecuencia. La base de datos se adapta al código y permite que nosotroso como desarrolladores se trabaje con un modelo orientado a objetos.

Desventajas: Puede ser más complejo si la base de datos es grande o está en producción. También puede ser menos adecuado cuando se necesita interactuar con bases de datos existentes.

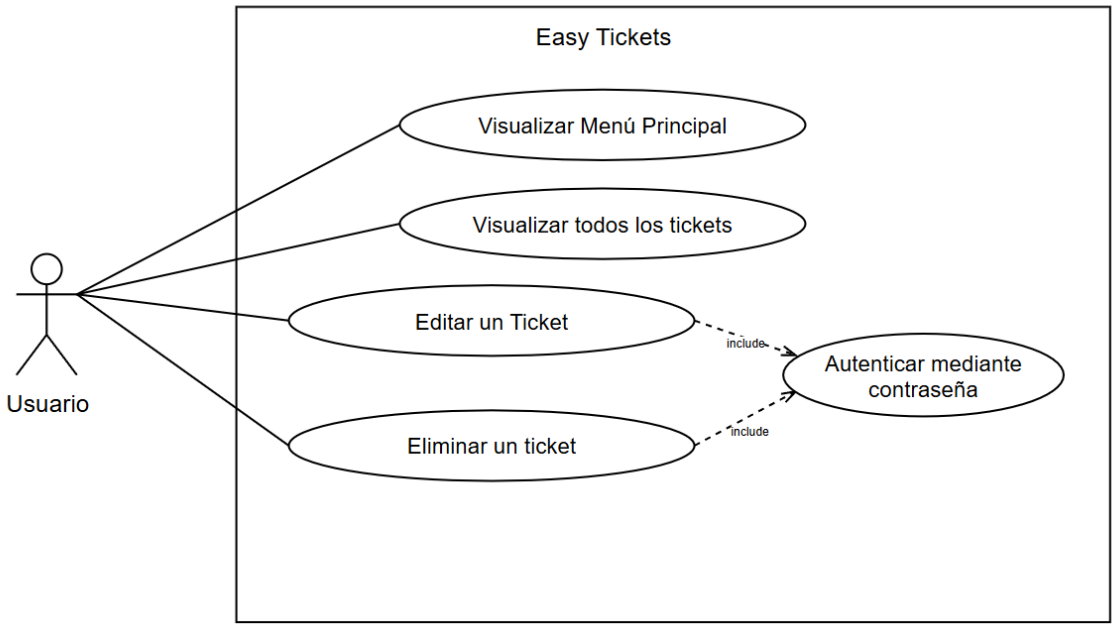
Enfoque 2: Database-First

En el enfoque Database-First, se parte de una base de datos ya existente, y Entity Framework genera automáticamente las clases de modelo en función de la estructura de la base de datos.

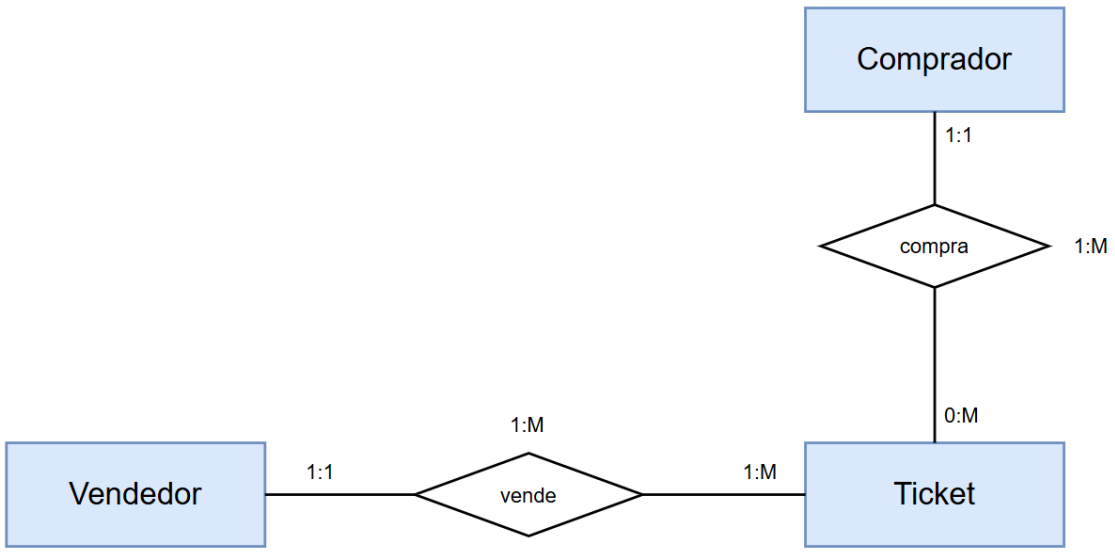
Ventajas: Es ideal cuando ya existe una base de datos con un esquema establecido o cuando se requiere mantener una estructura de base de datos específica.

Desventajas: Las modificaciones en el modelo deben realizarse directamente en la base de datos, y luego actualizar el código, lo que puede ser menos flexible y eficiente cuando se necesitan cambios grandes.

• Desarrolla el diagrama de caso de uso general que representa la solución al problema y lo describe de forma clara.



• Desarrolla el diagrama del modelo de datos.



• Desarrolla el prototipo de las principales interfaces de la aplicación web (mínimo 3).

URL: https://www.figma.com/proto/2Y1r21CNcIwbsBQCtsgw2R/Untitled?node-id=1-2&node-type=canvas&t=UwfzwrKs1R3YiBAF-0&scaling=min-zoom&content-scaling=fixed&page-id=0%3A1

Las herramientas seleccionadas son adecuadas y están bien justificadas para los siguientes motivos:

Visual Studio 2022:
Visual Studio es una de las IDEs más completas para el desarrollo de aplicaciones web en ASP.NET Core MVC. Con soporte para C#, la cual permite integrar fácilmente enfoques de desarrollo Code-First y Database-First. Visual Studio ofrece una amplia gama de herramientas de depuración y administración de bases de datos, lo cual facilita el desarrollo y la prueba de aplicaciones de manera eficiente. Su integración con sistemas de control de versiones, como Git, lo convierte en una elección sólida para desarrollar una aplicación.

Figma:

Figma es una excelente elección para crear prototipos interactivos y colaborativos. Para este proyecto, tambien Figma permite diseñar la interfaz gráfica de la aplicación de forma rápida e intuitiva.

Draw.io:

Draw.io es una herramienta práctica para la creación de diagramas, que será útil en la representación visual de la arquitectura del sistema, la base de datos, y los diagramas de casos de uso.

URLS SOLICITADOS

URL GITHUB: https://github.com/Bastian2704/ProyectoEasyTicket

URL VIDEO: Proyecto Progra