Área Académica de Ingeniería en Computadores

CE-3101 - Bases de Datos

Documentación Técnica

Profesor: Marco Rivera Meneses

Estudiantes:

David Robles Vargas

Carlos Andrés Mata Calderón

Felipe Vargas Jiménez

Jose Ignacio Calderón

José Umaña

II Semestre 2023

Índice

Links:

Diagramas:

Plan de Trabajo

Minutas

Bitacora

Estructuras Desarrolladas

Algoritmos Desarrollados

Problemas conocidos

Conclusiones

Recomendaciones

Bibliografía

Links:

Repositorio de GitHub:

https://github.com/Carlos12001/CineTEC.git

Plan de Trabajo Azure

https://dev.azure.com/felipevargas13/CineTEC2023/_backlogs/backlog/CineTEC2023%20Team/Features?showParents=true

Diagrama Conceptual

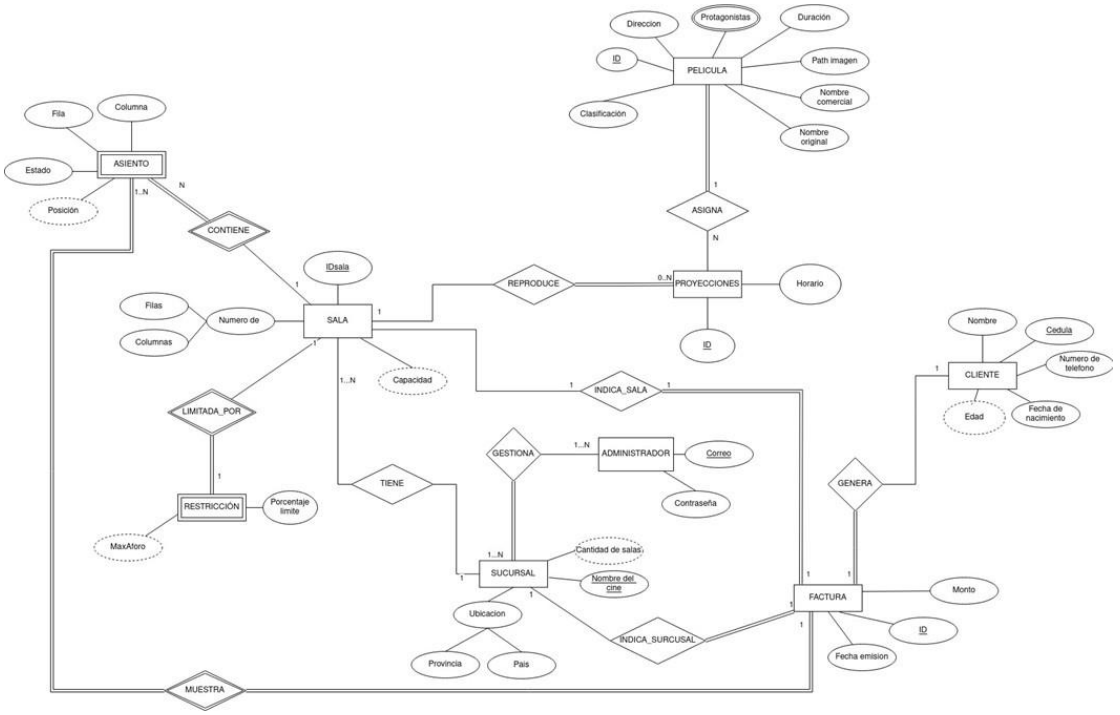


Diagrama Relacional

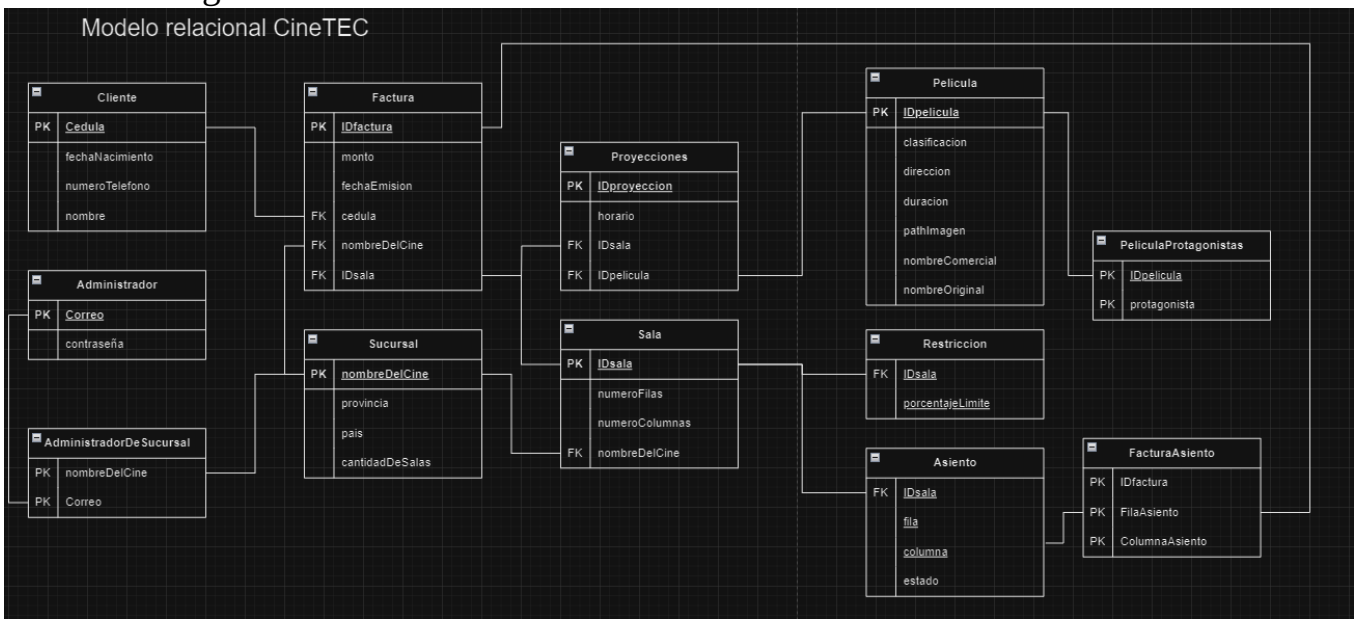


Diagrama de arquitectura

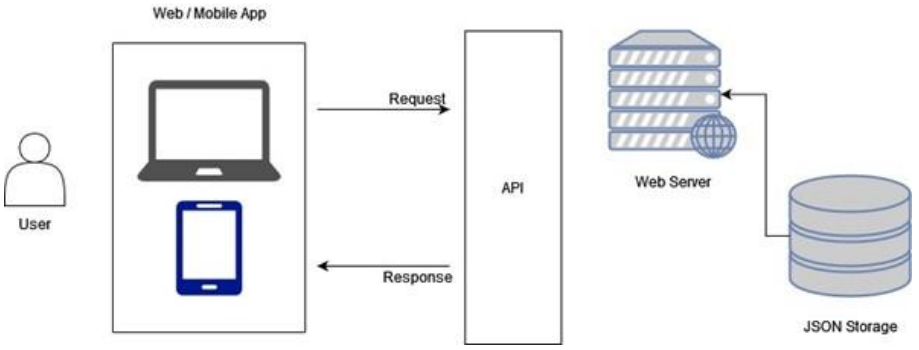
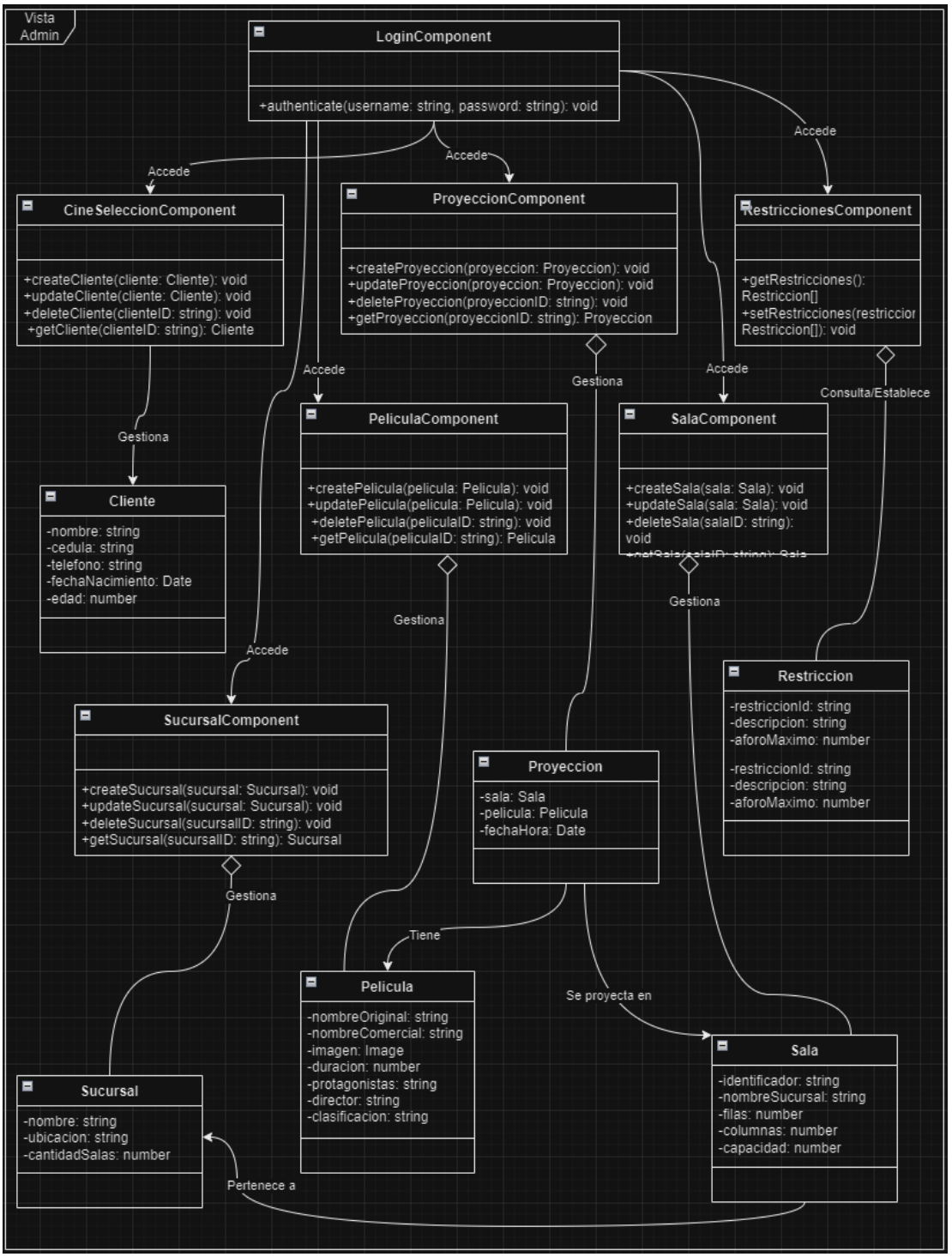
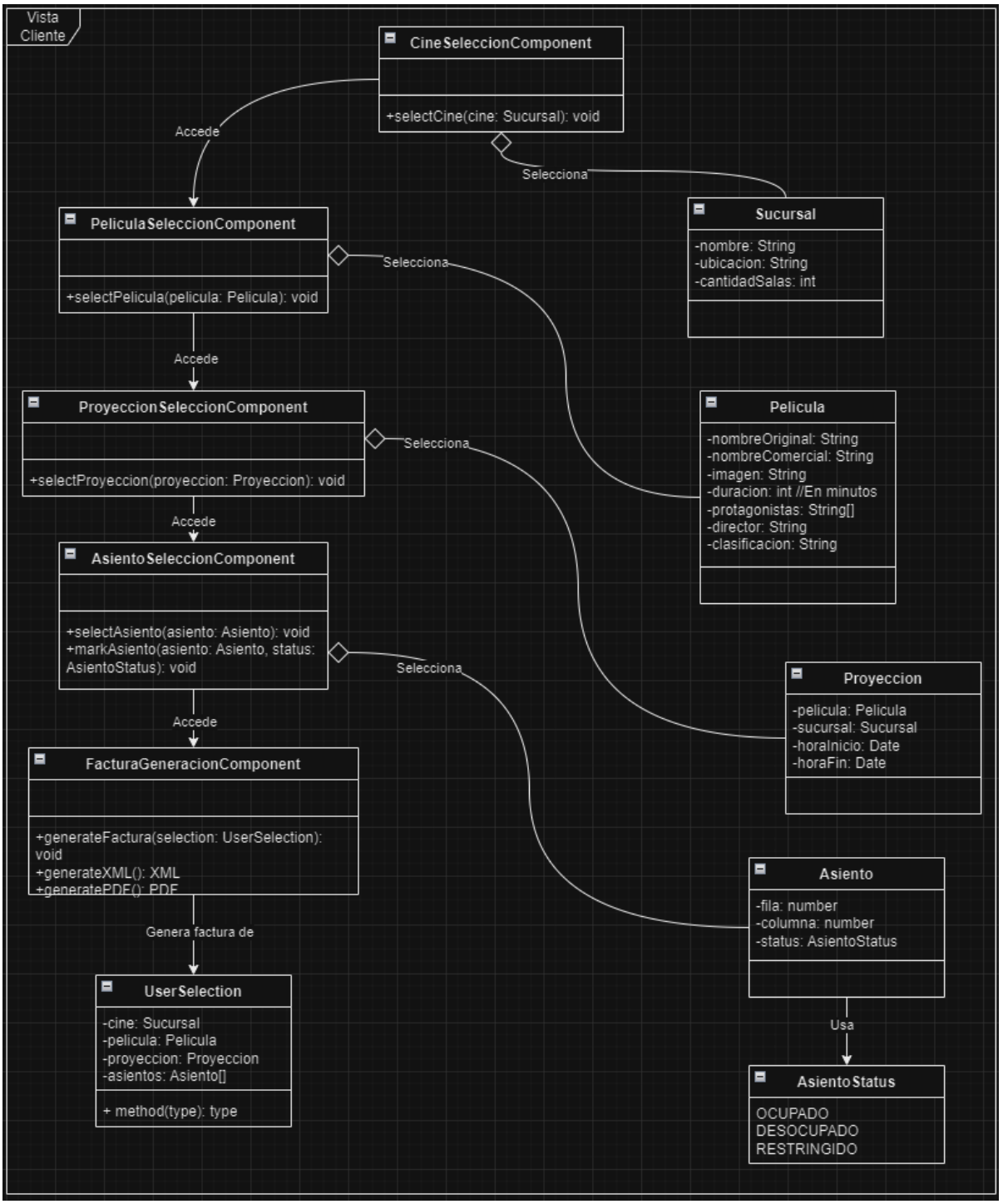
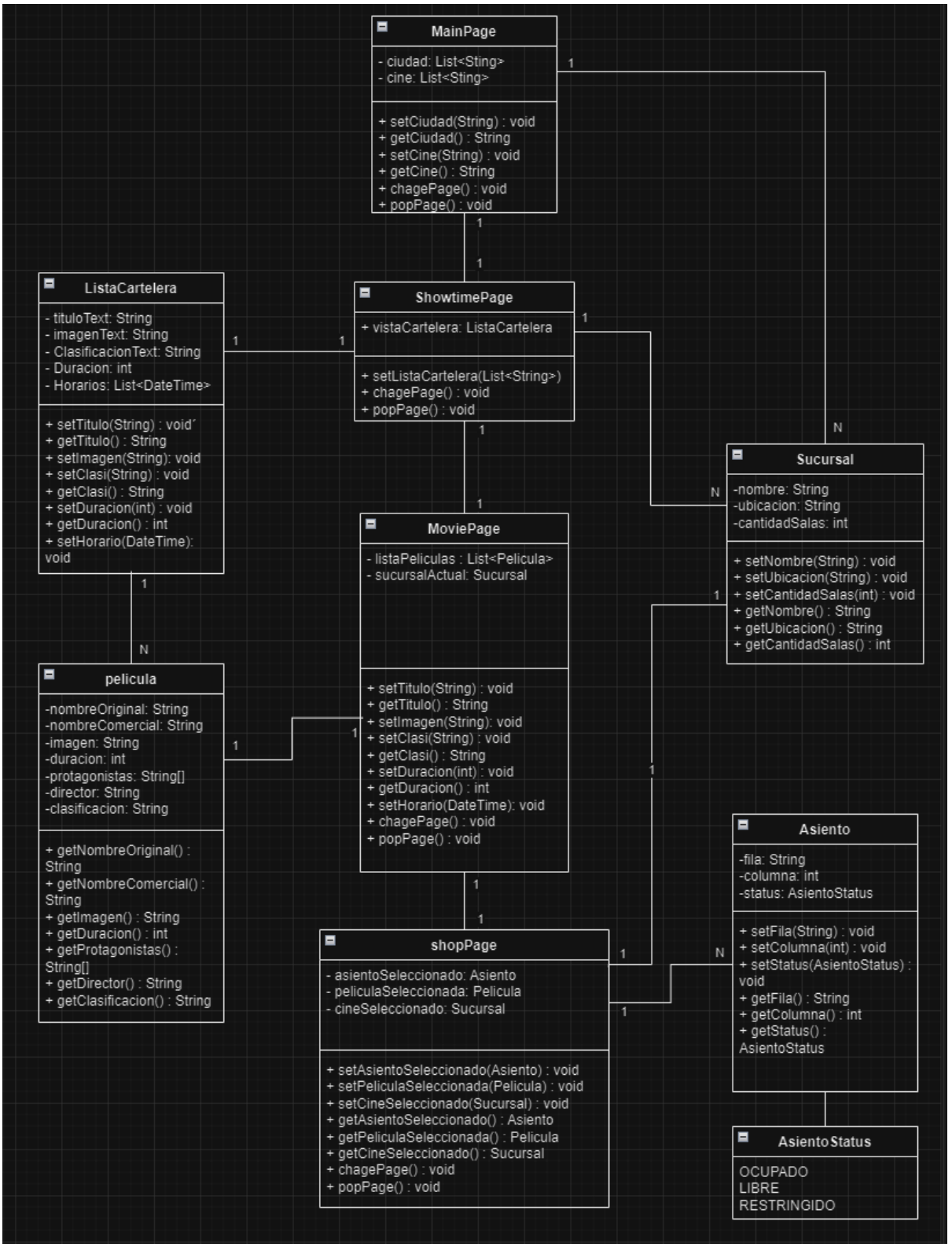


Diagrama de Clases


Vista Admin
LoginComponent
+authenticate(username: string, password: string): void
Accede
Accede
Accede
Accede
CineSeleccionComponent
+createCliente(cliente: Cliente): void
+updateCliente(cliente: Cliente): void
+deleteCliente(clienteID: string): void
+getCliente(clienteID: string): Cliente
ProyeccionComponent
+createProyeccion(proyeccion: Proyeccion): void
+updateProyeccion(proyeccion: Proyeccion): void
+deleteProyeccion(proyeccionID: string): void
+getProyeccion(proyeccionID: string): Proyeccion
RestriccionesComponent
+getRestricciones(): Restriccion[]
+setRestricciones(restriccior
Restriccion[]): void
Accede
Accede
Gestiona
Accede
Consulta/Establece
Gestiona
PeliculaComponent
+createPelicula(pelicula: Pelicula): void
+updatePelicula(pelicula: Pelicula): void
+deletePelicula(peliculaID: string): void
+getPelicula(peliculaID: string): Pelicula
SalaComponent
+createSala(sala: Sala): void
+updateSala(sala: Sala): void
+deleteSala(salaID: string): void
Gestiona
Gestiona
Accede
Cliente
-nombre: string
-cedula: string
-telefono: string
-fechaNacimiento: Date
-edad: number
Restriccion
-restriccionId: string
-descripcion: string
-aforoMaximo: number
-restriccionId: string
-descripcion: string
-aforoMaximo: number
SucursalComponent
+createSucursal(sucursal: Sucursal): void
+updateSucursal(sucursal: Sucursal): void
+deleteSucursal(sucursalID: string): void
+getSucursal(sucursalID: string): Sucursal
Proyeccion
-sala: Sala
-pelicula: Pelicula
-fechaHora: Date
Gestiona
Tiene
Pelicula
-nombreOriginal: string
-nombreComercial: string
-imagen: Image
-duracion: number
-protagonistas: string
-director: string
-clasificacion: string
Se proyecta en
Sala
-identificador: string
-nombreSucursal: string
-filas: number
-columnas: number
-capacidad: number
Sucursal
-nombre: string
-ubicacion: string
-cantidadSalas: number
Pertenece a


Vista Cliente
CineSeleccionComponent
+selectCine(cine: Sucursal): void
Accede
Selecciona
PeliculaSeleccionComponent
+selectPelicula(pelicula: Pelicula): void
Selecciona
Sucursal
-nombre: String
-ubicacion: String
-cantidadSalas: int
Accede
ProyeccionSeleccionComponent
+selectProyeccion(proyeccion: Proyeccion): void
Selecciona
Pelicula
-nombreOriginal: String
-nombreComercial: String
-imagen: String
-duracion: int //En minutos
-protagonistas: String[]
-director: String
-clasificacion: String
Accede
AsientoSeleccionComponent
+selectAsiento(asiento: Asiento): void
+markAsiento(asiento: Asiento, status: AsientoStatus): void
Selecciona
Proyeccion
-pelicula: Pelicula
-sucursal: Sucursal
-horaInicio: Date
-horaFin: Date
Accede
FacturaGeneracionComponent
+generateFactura(selection: UserSelection): void
+generateXML(): XML
+generatePDF(): PDF
Genera factura de
UserSelection
-cine: Sucursal
-pelicula: Pelicula
-proyeccion: Proyeccion
-asientos: Asiento[]
+ method(type): type
Asiento
-fila: number
-columna: number
-status: AsientoStatus
Usa
AsientoStatus
OCUPADO
DESOCUPADO
RESTRINGIDO


MainPage
- ciudad: List<Sting>
- cine: List<Sting>
+ setCiudad(String) : void
+ getCiudad() : String
+ setCine(String) : void
+ getCine() : String
+ chagePage() : void
+ popPage() : void
ListaCartelera
- tituloText: String
- imagenText: String
- ClasificacionText: String
- Duracion: int
- Horarios: List<DateTime>
+ setTitulo(String) : void´
+ getTitulo() : String
+ setImagen(String): void
+ setClasi(String) : void
+ getClasi() : String
+ setDuracion(int) : void
+ getDuracion() : int
+ setHorario(DateTime): void
ShowtimePage
+ vistaCartelera: ListaCartelera
+ setListaCartelera(List<String>)
+ chagePage() : void
+ popPage() : void
Sucursal
-nombre: String
-ubicacion: String
-cantidadSalas: int
+ setNombre(String) : void
+ setUbicacion(String) : void
+ setCantidadSalas(int) : void
+ getNombre() : String
+ getUbicacion() : String
+ getCantidadSalas() : int
MoviePage
- listaPeliculas : List<Pelicula>
- sucursalActual: Sucursal
+ setTitulo(String) : void
+ getTitulo() : String
+ setImagen(String): void
+ setClasi(String) : void
+ getClasi() : String
+ setDuracion(int) : void
+ getDuracion() : int
+ setHorario(DateTime): void
+ chagePage() : void
+ popPage() : void
pelicula
-nombreOriginal: String
-nombreComercial: String
-imagen: String
-duracion: int
-protagonistas: String[]
-director: String
-clasificacion: String
+ getNombreOriginal() : String
+ getNombreComercial() : String
+ getImagen() : String
+ getDuracion() : int
+ getProtagonistas() : String[]
+ getDirector() : String
+ getClasificacion() : String
Asiento
-fila: String
-columna: int
-status: AsientoStatus
+ setFila(String) : void
+ setColumna(int) : void
+ setStatus(AsientoStatus) : void
+ getFila() : String
+ getColumna() : int
+ getStatus() : AsientoStatus
shopPage
- asientoSeleccionado: Asiento
- peliculaSeleccionada: Pelicula
- cineSeleccionado: Sucursal
+ setAsientoSeleccionado(Asiento) : void
+ setPeliculaSeleccionada(Pelicula) : void
+ setCineSeleccionado(Sucursal) : void
+ getAsientoSeleccionado() : Asiento
+ getPeliculaSeleccionada() : Pelicula
+ getCineSeleccionado() : Sucursal
+ chagePage() : void
+ popPage() : void
AsientoStatus
OCUPADO
LIBRE
RESTRINGIDO
1
N

Mapeo para el diagrama relacional

Tipos de entidades fuertes

Estas fueron las entidades fuertes que se mapean

Administrador	
correo (PK)	contraseña

Película					
IDpelicula (PK)	clasificación	dirección	duración	pathimagen	nombreComercial

Sala					
IDsala (PK)	numeroFilas	dirección	duración	pathimagen	nombreComcercial

Proyecciones	
IDproyeccion (PK)	horario

Cliente			
cédula (PK)	fechaNacimiento	numeroTelefono	nombre

Sucursal			
nombreDelCine(PK)	provincia	país	cantidaDeSalas

Factura		
IDfactura(PK)	monto	fechaEmision

Tipos de entidades débiles

La entidad débil que se mapea es Asiento, se decide agregar como llave parcial el atributo fila y columna.

Asiento		
fila	columna	estado

Restricción
porcentajeLimite

Tipos de asociaciones binarias 1:1

Existen varias relaciones de 1 a 1 estas son

- Sala - Restricción

  Se decidió que cada restricción almacenera la cantidad de asientos que no seencuentran disponibles

Restricción	
IDsala (FK)	porcentaje Límite

Tipos de asociaciones binarias 1: N

- Relación sala-asiento

  Se identifica con el lado N la entidad asiento, este lado cuenta con participación total. Por lo tanto, se agrega a Asiento la llave primaria de Sala. Se obtiene la siguiente tabla:

Asiento			
IDsala (FK)	fila	columna	estado

- Relación sala-sucursal

  Se identifica con el lado N de la entidad Sala. Por lo tanto, se agrega a Sala la llave primaria de sucursal. Se obtiene la siguiente tabla:

Sala						
IDsala (PK)	numeroFilas	dirección	duración	pathimagen	nombreComcercial	nombreDelCine (FK)

- Relación sala-proyección

  Se identifica con el lado N la entidad Proyección, este lado cuenta con participación total. Por lo tanto, se agrega a Proyección la llave primaria de Sala. Se obtiene la siguiente tabla:

Proyecciones		
IDproyeccion (PK)	horario	IDsala (FK)

- Relación proyección-película

  Se identifica con el lado N de la entidad Proyección. Por lo tanto, se agrega aProyección la llave primaria de Película. Se obtiene la siguiente tabla:

Proyecciones			
IDproyeccion (PK)	horario	IDsala (FK)	IDpelicula (FK)

- Relación Factura-Sala

  Se identifica con el lado N de la entidad Factura. Por lo tanto, se agrega a Factura la llave primaria de Sala. La tabla resultante se observa de la siguiente manera.

Factura			
IDfactura(PK)	monto	fechaEmision	IDsala (FK)

- Relación Factura-Sucursal

  Se identifica con el lado N de la entidad Factura. Por lo tanto, se agrega a Factura la llave primaria de Sucursal. La tabla resultante se observa de la siguiente manera.

Factura				
IDfactura(PK)	monto	fechaEmision	IDsala (FK)	nombreDelCine (FK)

- Relación Factura-Cliente

  Se identifica con el lado N de la entidad Factura. Por lo tanto, se agrega a Factura la llave primaria de Cliente. La tabla resultante se observa de la siguiente manera.

Factura					
IDfactura(PK)	monto	fechaEmision	IDsala (FK)	nombreDelCine (FK)	cedula (FK)

Tipos de asociaciones binarias N:M

En el caso trabajado existen dos relaciones N a M:

- Relación Sucursal-Administrador

  Se creó una tabla como referencia cruzada en la cual se agregaron como llaves primarias las de ambas entidades.

Sucursal-Administrador	
nombreDelCine(PK)	**correo(PK)**

- Relación Factura-Asiento

  Igualmente se agregó una referencia cruzada con la llave primaria de factura y las llaves parciales de asiento pues esta es una entidad débil.

Factura-Asiento		
IDfactura(PK)	**FilaAsiento (PK)**	**ColumnaAsiento (PK)**

Atributos multivaluados

Como único atributo multivaluado en el caso, se encuentran los protagonistas de unapelícula, para lo que se creó una nueva relación, donde se incluye el atributo protagonista asociada a la llave primaria de la película.

- Protagonistas

PeliculaProtagonistas	
IDpelicula(PK)	**protagonista (PK)**

Descripción del modelo

Se crearon modelos detallados junto con sus atributos a partir de las relaciones establecidas en el diagrama relacional. Este proceso tiene como objetivo habilitar la conexión entre la base de datos y la aplicación ASP.NET mediante el uso de Entity Framework. Estos modelos representan de manera precisa las entidades de datos y sus características específicas.

Para gestionar la interacción con la base de datos, se implementó una clase de contexto de base de datos que actúa como un puente entre la aplicación y los datos almacenados. Esta clase permite llevar a cabo las operaciones CRUD necesarias para acceder y manipular la información en la base de datos. Además, se aplicaron estrategias de optimización, como el bundling y la minificación de archivos estáticos, con el propósito de mejorar el rendimiento de la aplicación al reducir la cantidad de solicitudes HTTP y el tamaño de los recursos estáticos.

Los controladores desempeñan un papel fundamental al gestionar las solicitudes HTTP entrantes y ejecutar acciones específicas. Estas acciones involucran interacciones con la base de datos a través del contexto de Entity Framework para proporcionar respuestas, ya sean vistas HTML o datos JSON, a los clientes de la aplicación. En conjunto, esta arquitectura proporciona una base sólida para desarrollar aplicaciones web de manera eficiente y con capacidad de escalabilidad.

Plan de Trabajo

Metas

● Desarrollar Funcionalidades
    ○ Programación del API/REST Service en C#-.NET, incluyendo métodos de POST y GET. ○ Implementación de la interfaz por parte vista del cliente y administrador en Angular, utilizando Bootstrap, HTML5 y CSS3.
    ○ Desarrollo de una aplicación móvil en Flutter, para desplegar la interfaz vista por el cliente en el móvil.
    ○ Conexión del API/REST Service con la web app en Angular.
    ○ Conexión del API/REST Service con la web app en Flutter.
● Construir Base de Datos ○ Diseño y estructuración de la base de datos en formatos Json o XML.
● Diseñar Arquitectura ○ Creación de diagramas y modelos conceptuales y relacionales.
    ○ Descripción técnica de la estructura del sistema, desde la base de datos hasta la interfaz del usuario.
● Instalar y Configurar ○ Despliegue y configuración de la aplicación en un ambiente Windows en el IIS.
    ○ Generación de guías detalladas para facilitar futuras instalaciones.

Roles

Front End Manager: Se encarga de llevar un control de las tareas a realizar y posibles adiciones que se pueden necesitar a la hora del uso de la aplicación y la web. Además, coordina con los BackEnd Developer y el Mobile Developer acerca de la conexión.
Front End Designer: Crea mediante código la interfaz e implementa las funcionalidades necesarias para una sencilla utilización de la página.
Back End API Developer: Responsable de crear y mantener la interfaz de programación de la API, permitiendo una buena comunicación entre las aplicaciones.
Back End DataBase Designer: Se encarga de diseñar, implementar y mantener la estructura de la base de datos donde se almacenan y gestionan los datos del sistema.
Mobile Developer: Se encarga de diseñar y crear la interfaz gráfica para mostrar al cliente. Además, implementa toda funcionalidad necesaria con el fin que la app sea amigable con el usuario.

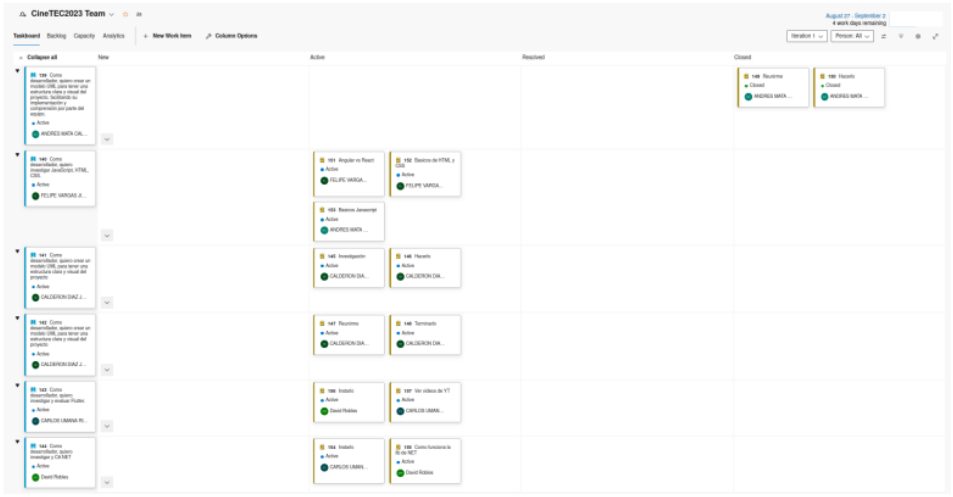
Reglas

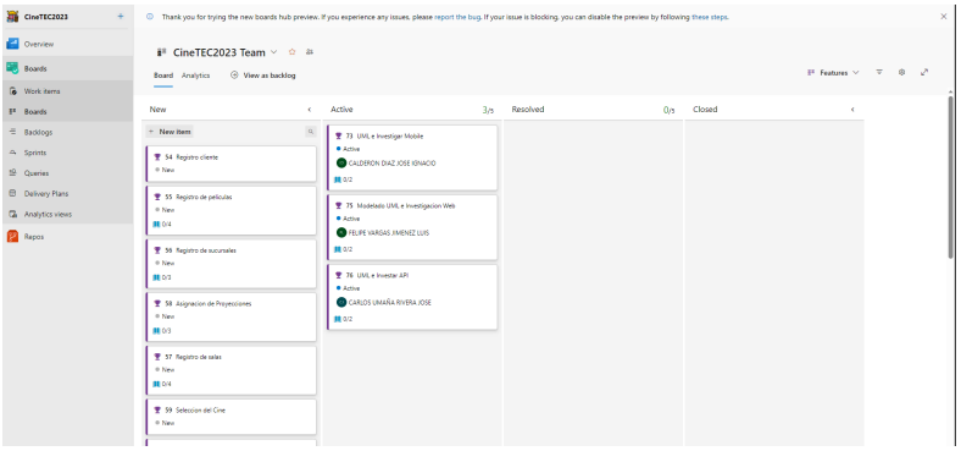
● Las reuniones sincrónicas se harán en su mayoría mediante discord, las cuales se emplearán para definir detalles específicos de las tareas a realizar. En el caso de ser posible o necesario, se harán reuniones presenciales.
● La comunicación restante se dará mediante un grupo de WhatsApp, en su mayoría para presentar actualizaciones en el desarrollo, y notificar de cualquier problema encontrado al resto del equipo.
● El control de versiones del código se hará mediante GitHub, con el fin de que todos los miembros del grupo tengan acceso al código actualizado por otros miembros.
● El código se realizará y comentará en el idioma inglés con el fin de mantener la coherencia del desarrollo.
● Los comentarios de GitHub se utilizarán usando la estandarización de Conventional Commits, por lo que también se comenta en inglés.
● 2 personas no pueden trabajar sobre el mismo Branch. Cada developer tiene su propio branch en el cual trabaja e imprenta sus características.
● Nunca comentar en el master en el GitHub.
● Para hacer merge se crea un branch de unión para unificar el código.
● Comentar al estilo javadocs.
● Nombrar funciones y variables lo más significativo y corto posible. (Al estilo clean code).
● Mantener al día el Azure con las tareas y user stories implementadas.

Cronograma

Azure: https://dev.azure.com/felipevargas13/CineTEC2023

Primer Sprint 1 (27 agosto - 2 sept)





Minutas

Minutas de las sesiones de trabajo

Para el Sprint 1#

Num	*Participantes*	*Fecha -Hora Inicio*	*Fecha-Hora Finalizado*	*Acuerdos*	*Desacuerdos*
1	*Todos los integrantes - De forma PRESENCIAL*	***25/08/2023*** *2 pm*	***25/08/2023*** *4:45 pm*	*Lectura del proyecto y los lineamientos de trabajo, así como una posible solución de los problemas encontrados en el momentos*	*No hay desacuerdos*
2	*Todos los integrantes de forma VIRTUAL por medio de la plataforma discord*	***26/08/2023*** *6 pm*	***26/08/2023*** *9 pm*	*Creación del diagrama entidad relación en Drawio y la división de lso roles de trabajo al igual que la división del diagrama UML para su trabajo en grupos de trabajo correspondientes*	*Hay breves discusiones sobre cómo crear dicho modelo pero de acuerdo a las conversaciones entre los estudiantes y las discusión se llegaron a acuerdos.*

Minutas de las sesiones de trabajo
para el sprint #2

Num	*Participantes*	*Fecha -Hora Inicio*	*Fecha-Hora Finalizado*	*Acuerdos*	*Desacuerdo s*
3	Felipe Vargas y Igancio Calderon	***05/09/2023 8 PM***	***05/09/2023 11:40 PM***	Creación del diseño estético de la web así como de la aplicación móvil	Hay desacuerdos sobre el diseño de la web. Por el momento, se lograron crear intermediarios. Se debe conversar con el resto del equipo para llegar a un acuerdo final
4	Carlos Calderon, David Robles, José Umaña	***05/09/2023 7 PM***	***05/09/2023 11:00 PM***	Revisión en avances de bases de datos, además de que Jose y Carlos estuvieron analizando posibles soluciones de implementación para la conexión de la web con el server	No hay desacuerdos

Minutas de las sesiones de trabajo
para el sprint #3

Num	*Participantes*	*Fecha -Hora Inicio*	*Fecha-Hora Finalizado*	*Acuerdos*	*Desacuerdo s*
5	Todos los participantes	***11/09/2023 7:50 PM***	***12/09/2023 1:53 PM***	Creación y discusión de los diagramas tanto el relacional como el de arquitectura al igual que mostrar avance en la aplicación Web, en la base de datos y en la aplicación móvil	Hay breves desacuerdos de acuerdo al avance de cada uno de los participantes. Se va a trabajar con un cronograma para mostrar avances más notorios y con pruebas.

Bitácora

Usuario: Carlos12001

Hash: a133406
Fecha: 2023-09-21
Mensaje: test: all
Hash: 92cc1b5
Fecha: 2023-09-21
Mensaje: style: add style login
Hash: b6622f9
Fecha: 2023-09-21
Mensaje: chore: delte client button
Hash: 5e30fa4
Fecha: 2023-09-21
Mensaje: feat: editor room finish
Hash: a6076a0
Fecha: 2023-09-21
Mensaje: feat: cliente
Hash: cf07b1e
Fecha: 2023-09-21
Mensaje: fix: cinema string room id
Hash: e47e89b
Fecha: 2023-09-20
Mensaje: Merge branch 'main' of https://github.com/Carlos12001/CineTEC
Hash: 6f45fc1
Fecha: 2023-09-20
Mensaje: feat: delete movies finish
Hash: 0189f5f
Fecha: 2023-09-20
Mensaje: feat: add movies
Hash: 12243b9
Fecha: 2023-09-20
Mensaje: feat: create movie
Hash: 6176024
Fecha: 2023-09-20
Mensaje: feat: add other atributes
Hash: fec0c00
Fecha: 2023-09-20
Mensaje: fix: editing without edit all
Hash: 62ca173
Fecha: 2023-09-20
Mensaje: feat: add editor movies broken
Hash: 290dc14
Fecha: 2023-09-20
Mensaje: feat: no entity
Hash: db7033c
Fecha: 2023-09-20
Mensaje: feat: finished the slide bar
Hash: 84650f7
Fecha: 2023-09-19
Mensaje: fix: things
Hash: 588ae93
Fecha: 2023-09-19

Mensaje: feat: todofunciona
Hash: 9c43671
Fecha: 2023-09-19
Mensaje: Merge branch 'admin_editor' into merge_2_api_objetcs_and_web_admin_editor
Hash: ac24556
Fecha: 2023-09-19
Mensaje: pichassas
Hash: 49fc41c
Fecha: 2023-09-19
Mensaje: feat: show editor
Hash: bdfd1ba
Fecha: 2023-09-19
Mensaje: chore: atributes
Hash: 22d259b
Fecha: 2023-09-18
Mensaje: feat: add gretting admin editor
Hash: 853d70d
Fecha: 2023-09-18
Mensaje: fix: admin login post
Hash: 944cf24
Fecha: 2023-09-18
Mensaje: feat: add ignore
Hash: 3d68f39
Fecha: 2023-09-18
Mensaje: fix: reapair conecction with api
Hash: e297281
Fecha: 2023-09-17
Mensaje: Merge remote-tracking branch 'origin/admin_view' into merge_test_login_api_web
Hash: 9691848
Fecha: 2023-09-17
Mensaje: feat: idea for admin editor
Hash: b22607f
Fecha: 2023-09-17
Mensaje: fix: conecction admin login
Hash: d29abf7
Fecha: 2023-09-17
Mensaje: feat: add the admin editor
Hash: 26d4039
Fecha: 2023-09-15
Mensaje: build(vscode): configure Firefox for Angular debugging
Hash: 0dc4bda
Fecha: 2023-09-15
Mensaje: feat: change style home
Hash: cbe743c
Fecha: 2023-09-14
Mensaje: feat: initialize web finish
Hash: 7394900
Fecha: 2023-09-14
Mensaje: feat: add fotter
Hash: 6056aa0
Fecha: 2023-09-12
Mensaje: feat: route home page
Hash: 5ed6cf7

Fecha: 2023-09-12
Mensaje: chore: delete defaults things
Hash: d3561e4
Fecha: 2023-09-12
Mensaje: feat: inittialize angular project
Hash: 8181af8
Fecha: 2023-09-04
Mensaje: docs: update README.md
Hash: 5709723
Fecha: 2023-09-04
Mensaje: docs: diagrams and spefication
Hash: 473c763
Fecha: 2023-09-04
Mensaje: docs: add readmes

Usuario: Felipe Vargas
Hash: 26c2887
Fecha: 2023-09-21
Mensaje: Merge branch 'mobileWinternet'
Hash: 914b6c5
Fecha: 2023-09-21
Mensaje: docs: internal documentation web
Hash: 7413d42
Fecha: 2023-09-21
Mensaje: feat: projections working
Hash: dfb5487
Fecha: 2023-09-21
Mensaje: feat: show cinemas
Hash: c4369c9
Fecha: 2023-09-21
Mensaje: chore: reformart code
Hash: e3f152f
Fecha: 2023-09-21
Mensaje: style: set global styles
Hash: 1465b74
Fecha: 2023-09-21
Mensaje: feat: cinema estrucutre
Hash: 926badb
Fecha: 2023-09-20
Mensaje: chore: movies
Hash: 00e4598
Fecha: 2023-09-20
Mensaje: Merge remote-tracking branch 'origin/mobileWApi' into merge_mobile_final_and_editor_admin
Hash: 84e83a2
Fecha: 2023-09-20
Mensaje: feat: updater broken
Hash: 32af745
Fecha: 2023-09-20
Mensaje: feat: services get movies admin editor finish
Hash: 7774094
Fecha: 2023-09-20

Mensaje: chore: dont show new movie delete
Hash: 81ab22b
Fecha: 2023-09-20
Mensaje: style: fixing
Hash: a7a1c72
Fecha: 2023-09-20
Mensaje: style: reapair the style editor
Hash: fc17fbf
Fecha: 2023-09-20
Mensaje: chore: clean example
Hash: c5ba395
Fecha: 2023-09-20
Mensaje: feat: add delete button
Hash: 550f42e
Fecha: 2023-09-20
Mensaje: fix: little problems
Hash: 40970b3
Fecha: 2023-09-20
Mensaje: style: repair stytle font size movies
Hash: 606203e
Fecha: 2023-09-20
Mensaje: chore: cleaning movie
Hash: 6ffa15d
Fecha: 2023-09-19
Mensaje: feat: fsadfgfsg
Hash: 3222904
Fecha: 2023-09-19
Mensaje: fix: project
Hash: fcd3d18
Fecha: 2023-09-19
Mensaje: delte
Hash: 11b132e
Fecha: 2023-09-19
Mensaje: Actualizado .gitignore y archivos desrastreados
Hash: 998bece
Fecha: 2023-09-19
Mensaje: feat: all working
Hash: 031224a
Fecha: 2023-09-19
Mensaje: FUNCIONA
Hash: 9deb369
Fecha: 2023-09-19
Mensaje: Actualizado .gitignore y archivos desrastreados
Hash: 8dd2ab2
Fecha: 2023-09-19
Mensaje: feat: banner buttones editor admin
Hash: b634c3f
Fecha: 2023-09-19
Mensaje: Merge branch 'admin_editor' into mergeMobileApiWeb
Hash: 89beb1e
Fecha: 2023-09-18
Mensaje: chore: limpieza
Hash: 91ee88b

Fecha: 2023-09-18
Mensaje: chore: delte test
Hash: 8041a86
Fecha: 2023-09-17
Mensaje: feat: admin login
Hash: 68b777b
Fecha: 2023-09-15
Mensaje: chore: formart
Hash: 2334fd1
Fecha: 2023-09-15
Mensaje: feat: create admin-login component
Hash: b92300b
Fecha: 2023-09-15
Mensaje: fix: delete links home
Hash: a12df65
Fecha: 2023-09-15
Mensaje: feat: home page finished
Hash: 2a56229
Fecha: 2023-09-14
Mensaje: feat: add header
Hash: 4974077
Fecha: 2023-09-14
Mensaje: feat: set default config
Hash: 2fca2a4
Fecha: 2023-09-05
Mensaje: docs: delte diagram
Hash: 1c00fd9
Fecha: 2023-09-05
Mensaje: Added Comunication Diagram.drawio

Usuario: Ignacio Calderón
Hash: 01367db
Fecha: 2023-09-21
Mensaje: docu
Hash: 0bd214e
Fecha: 2023-09-21
Mensaje: final version
Hash: 91a1c4b
Fecha: 2023-09-21
Mensaje: Merge branch 'main' into mobileWinternet
Hash: 503c821
Fecha: 2023-09-21
Mensaje: change obj class
Hash: 9b11068
Fecha: 2023-09-21
Mensaje: docu
Hash: d0d0379
Fecha: 2023-09-20
Mensaje: add cinema DB
Hash: 898bbfd
Fecha: 2023-09-20
Mensaje: fix small deatils

Hash: 0aafc6f
Fecha: 2023-09-20
Mensaje: add seats selection page
Hash: 2fd2757
Fecha: 2023-09-19
Mensaje: add movie details
Hash: 33f6919
Fecha: 2023-09-18
Mensaje: movie theather selection
Hash: 7b36ebc
Fecha: 2023-09-18
Mensaje: add showtime
Hash: 754f5db
Fecha: 2023-09-17
Mensaje: reformat
Hash: a2b715d
Fecha: 2023-09-17
Mensaje: home page
Hash: 26c463f
Fecha: 2023-09-11
Mensaje: add app
Hash: 2e77d34
Fecha: 2023-09-11
Mensaje: add first flutter app

Usuario: David Robles
Hash: cc3a116
Fecha: 2023-09-20
Mensaje: feat: javadocs like documentation for most of the methods utilized.
Hash: 59df701
Fecha: 2023-09-19
Mensaje: fix: fixed projection id to string (from integer).
Hash: 05cb6fe
Fecha: 2023-09-18
Mensaje: feat: added projections controller and post/delete methods changed
Hash: ed687ab
Fecha: 2023-09-17
Mensaje: Update .gitignore
Hash: 5178597
Fecha: 2023-09-17
Mensaje: Merge branch 'WebApi' of https://github.com/Carlos12001/CineTEC into WebApi
Hash: 6ab750f
Fecha: 2023-09-17
Mensaje: fix: changed properties in movies and post method
Hash: 4d302bb
Fecha: 2023-09-17
Mensaje: feat: get for specific item
Hash: 6546a30
Fecha: 2023-09-17
Mensaje: fix: Just some name corrections
Hash: e414793
Fecha: 2023-09-17

Mensaje: feat: room and movie controllers
Hash: 028ae3a
Fecha: 2023-09-17
Mensaje: feat: functional post and delete methods for admin
Hash: ceddd93
Fecha: 2023-09-05
Mensaje: Actualizar Comunication Diagram.drawio
Hash: 98d586c
Fecha: 2023-09-05
Mensaje: Actualizar Comunication Diagram.drawio

Usuario: Jose
Hash: d4fcb6d
Fecha: 2023-09-19
Mensaje: fix: project
Hash: ca12094
Fecha: 2023-09-17
Mensaje: feat: cinema data and controller
Hash: fab1981
Fecha: 2023-09-16
Mensaje: feat: more entities
Hash: 1c09a9f
Fecha: 2023-09-14
Mensaje: feat: api admin model controller

Estructuras desarrolladas

Para tener un mejor manejo de los datos se siguió el modelo relacional tanto para la base de datos como para la página web, también se hizo uso de controllers para casos específicos como administrador, cinema, home, movie, projections, Room estos controllers responden a las solicitudes de la página de inicio para realizar la solicitud, en cuanto a la parte de almacenamiento en la base de datos se trabajó con JSON para poder ingresar y realizar la solicitud de los datos ingresados anteriormente al igual que si se requieren escribir nuevos datos en esta.

Admin
string: id
string: password
string: name
string: email

Cinema
string: name
string: province
string: country
int: roomsamount
List(int): roomid

Movie
string: rating
string: director
string: duration
string: image
string: cname
string: oname
List(string): prota

Projections
int: id
Datetime: horary
int: roomid
string: movieid

Room

int: id
int: rows
int: colums
string: treathername
List(int): projectionid

Es importante destacar que para la implementación no se lograron realizar todos los modelos como están descritos en el diagrama relacional, pero se logró realizar una conexión entre la API y la web específicamente en el administrador y sus funciones que posee por lo que se siguió la misma estructura de las relaciones, sus Primary key y Foreing Key. Además, se realiza la conexión entre la API y la aplicación móvil, se logran extraer los cines y películas.

Algoritmos desarrollados

Algoritmo de login

En el lado de la app web, se verifica el inicio de sesión ya sea de un empleado o cliente, de forma tal que se realiza un request de tipo GET que envía el *username* y *password* del usuario que quiere ingresar al sistema. Dentro del API se realiza el análisis, comparando estos valores con la tabla respectiva para cada entidad (empleado o cliente).
Si se logra tener una congruencia entre nombre y contraseña se le permite al usuario ingresar a las demás vistas de la aplicación, en caso contrario enviará un mensaje de "Nombre de usuario o contraseña incorrectos".

Algoritmo de creación de tablas de ítems

Para llenar el componente item-holder con cada uno de los ítems para cierta categoría (projections, cinema, etc) se hace una llamada de GET según el URL.
La respuesta después es introducida en un ciclo que genera un componente ítem dentro de item-holder con los datos de cada ítem.
El componente ítem posee una variable que identifica a cada posible item que puede contener, sin embargo, sólo mostrará aquellas que estén condicionadas para el URL donde se encuentre el usuario.

Algoritmo de servicios

Para el proceso de sincronización con el API se crearon ciertas funciones que permiten actualizar cada una de las tablas que se necesitarán en la aplicación móvil, como lo son la tabla de clientes, películas, salas y proyecciones. Cuando se desee acceder a algunas de las vistas que necesiten la información de algunas de estas tablas, se procederá a actualizar, realizando las consultas necesarias al API y agregándola a la base de datos, la información necesaria para mostrar la versión más actualizada.
Además, se crea una opción para actualizar la base de datos en el momento en el momento que se desee.

Problemas Conocidos

Conexión móvil-API

Al utilizar la aplicación móvil, no es posible acceder a la API que se ejecuta en un servidor local, lo que impide que el emulador o el teléfono puedan conectarse a ella. Para resolver este problema, se opta por usar ngrok, una herramienta que expone servidores locales a internet mediante túneles seguros. Sin embargo, a pesar de que esta solución permite que la aplicación móvil acceda a la API, su uso no resulta del todo cómodo, ya que es necesario modificar la dirección HTTP del servidor en el código cada vez que este se inicia

Conclusiones

1- La mayoría de los sistemas de reporte necesitan de NuGet packages disponibles en las bibliotecas de .NET de Visual Studio.
2- Es posible desarrollar aplicaciones Web bastante robustas con Angular ya que posee una gran cantidad de funcionalidades.
3- Para agilizar el desarrollo sin necesidad de instalar ambientes u otras aplicaciones es posible trabajar con Azure y otras plataformas que permiten subir sitios web gratuitos.
4- El entorno de desarrollo de Android para desarrollar una aplicación móvil permite crear aplicaciones móviles de manera sencilla y eficiente.

Recomendaciones

1- Se recomienda utilizar el entorno de desarrollo Android para desarrollar aplicaciones móviles por las diferentes facilidades y funcionalidades que ofrece.
2- Es recomendable que, al desarrollar un API en .NET, utilizar Visual Studio para tener acceso a mayor funcionalidad como sistemas de reporte, aunque es necesario realizar una lectura intensiva para lograr entender conceptos generales y detallados sobre funciones específicas.
3- Antes de comenzar un proyecto con Angular es importante hacer una investigación inicial puesto que es una plataforma con un grado de complejidad importante, producto de sus muchas funcionalidades.

Bibliografía

- Bootstrap Themes & Templates (2022-02-25). Recuperado de: https://wrapbootstrap.com/
- How to Write Doc Comments for the Javadoc Tool. (2022-02-25). Recuperado de: http://www.oracle.com/technetwork/articles/java/index-137868.html
- C# Coding Conventions (C# Programming Guide). (2022-02-25). Recuperado de: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/inside-a-
- program/coding-conventions
- Anexos y estructuras (Ministerio de Hacienda). (2022-02-25). Recuperado de: https://www.hacienda.go.cr/ATV/ComprobanteElectronico/frmAnexosyEstructuras.aspx#
- Angular https://angular.io/docs