Integrantes:

-Vanessa Susaya Cardenas

-Joaquín de la Riva Aysanoa

-Guillermo Garnica Valencia

Atlantic City

Transformación digital para una gestión de sistemas con enfoque data-driven y personalizada

Contenido

[Introducción al proyecto 2](#_Toc195009653)

[Contexto del proyecto 2](#_Toc195009654)

[Propósito y objetivos 2](#_Toc195009655)

[Impacto esperado en la organización 3](#_Toc195009656)

[Análisis de necesidades y requerimientos 4](#_Toc195009657)

[Descripción del problema actual 4](#_Toc195009658)

[Identificación de Stakeholders 4](#_Toc195009659)

[Requerimientos funcionales y no funcionales 5](#_Toc195009660)

[Diseño del sistema 6](#_Toc195009661)

[Arquitectura del sistema 6](#_Toc195009662)

[Diagrama de componentes 6](#_Toc195009663)

[Diseño de la base de datos 7](#_Toc195009664)

[Prototipo de interfaz de usuario 9](#_Toc195009665)

[Flujo de usuario y funcionalidades 11](#_Toc195009666)

[Desarrollo y funcionalidad 13](#_Toc195009667)

[Visual Studio Code 13](#_Toc195009668)

[Base de datos MySQL Workbench 13](#_Toc195009669)

[Node.js 14](#_Toc195009670)

[Implementación del Frontend 14](#_Toc195009671)

[Implementación del Backend 16](#_Toc195009672)

[Conexión a la base de datos 18](#_Toc195009673)

[Estructura de archivos 19](#_Toc195009674)

[Presentación final del sistema 20](#_Toc195009675)

[Reflexión sobre el proceso de desarrollo 20](#_Toc195009676)

[Conclusiones y recomendaciones 21](#_Toc195009677)

[Anexos 21](#_Toc195009678)

[Código fuente 21](#_Toc195009679)

[Materiales visuales 21](#_Toc195009680)

[App.js 21](#_Toc195009681)

[Db.js 22](#_Toc195009682)

[Ejemplo de CRUD en clientes.js 22](#_Toc195009683)

# Introducción al proyecto

## Contexto del proyecto

Atlantic City, es una empresa que se desarrolla en el sector del entretenimiento, ha experimentado un crecimiento significativo en los últimos años, lo que ha impulsado sus operaciones en diversas áreas y el aumento de su plantilla a más de 1,200 empleados. Sin embargo, este crecimiento no ha ido acompañado de una actualización en su infraestructura tecnológica, lo que ha generado serios inconvenientes en la gestión de clientes.

Actualmente la empresa utiliza múltiples sistemas y aplicaciones para manejar distintos aspectos de la relación con sus clientes, tales como inscripciones, seguimiento de promociones, gestión de membresías y operaciones diarias. La falta de integración entre estos sistemas ocasiona duplicidad de información y una coordinación deficiente entre departamentos. Esta situación repercute directamente la atención al cliente, que resulta poco personalizada y a menudo inconsciente, además de limitar la capacidad de la empresa para tomar decisiones estratégicas basadas en datos confiables y en tiempo real. Ante este escenario, se evidencia la necesidad de modernizar el sistema de gestión de clientes, centralizando la información y adoptando un enfoque data-driven que potencie tanto la eficiencia operativa como la experiencia del cliente.

## Propósito y objetivos

Propósito

El propósito del sistema es desarrollar una plataforma integral de gestión de clientes que transforme la operatividad de Atlantic City, permitiendo la centralización y estructuración de la información, la automatización de procesos internos y la toma de decisiones basada en datos en tiempo real. Este sistema busca no solo optimizar la eficiencia operativa y la organización de registros, si no también mejorar significativamente la experiencia del cliente mediante una interfaz intuitiva y segura.

Objetivos generales

Desarrollar un sistema de información de gestión de clientes que optimice la operación de Atlantic City, mejorando la experiencia del cliente a través de la centralización de datos, la automatización de procesos y la implementación de estrategias basadas en datos para la toma de decisiones

* *Desarrollar un sistema de gestión de datos* que permita a los empleados registrar, actualizar y consultar información de manera eficiente, asegurando que los procesos sean rápidos y accesibles.
* *Optimizar la administración de la información* mediante una base de datos bien estructurada, minimizando redundancias, mejorando la organización de los registros y garantizando la integridad de los datos almacenados.
* *Facilitar la interacción del empleado* con el sistema a través de una interfaz gráfica intuitiva que simplifique la introducción, edición y consulta de datos sin requerir el uso de comandos técnicos o acceso directo a las bases de datos.
* *Garantizar la seguridad y confiabilidad* del sistema estableciendo mecanismos de autenticación y control de permisos para evitar accesos no autorizados.

Objetivos Específicos:

* *Desarrollar una base de datos estructurada* que almacene la información de manera eficiente, definiendo las tablas, relacionales y restricciones necesarias para mantener la organización y evitar datos duplicados o inconsistentes.
* *Implementar relaciones entre los datos* para optimizar la consulta y actualización de información, garantizando que el sistema pueda gestionar múltiples registros vinculados sin afectar el rendimiento.
* *Asegurar la integridad de los datos* mediante la implementación de restricciones como claves primarias, claves foráneas y reglas de validación, evitando registros erróneos o información incompleta dentro del sistema
* *Optimizar los procesos internos del sistema* reduciendo la necesidad de intervención manual de tareas repetitivas
* *Mejorar la rapidez y precisión en tiempo real* asegurando que cualquier modificación en los registros se refleje de inmediato a los usuarios con acceso autorizado.
* *Diseñar un sistema de navegación intuitivo* que organice las funciones y módulos de manera lógica, reduciendo la curva de aprendizaje y permitiendo que cualquier empleado pueda operar el sistema sin dificultades

## Impacto esperado en la organización

El impacto que se espera en la organización se resume en:

Mejora en la eficiencia operativa:

Al centralizar y estructurar la información de clientes, se eliminarán procesos redundantes y agilizarán tareas diarias, lo que se traducirá en una operación más rápida y coordinada.

Atención al cliente personalizada y de calidad:

Con una base de datos unificada, los empleados podrán acceder a la información completa y actualizada de cada cliente, permitiendo ofrecer servicios y promociones, adaptados a sus necesidades lo que incrementará la satisfacción y fidelidad del cliente.

Toma de decisiones basadas en datos:

La integración de datos en tiempo real y la generación de reportes y análisis facilitaran la identificación de oportunidades y áreas de mejora, permitiendo a la gerencia tomar decisiones estratégicas fundamentadas en información precisa.

Optimización de recursos humanos:

La automatización de procesos repetitivos liberará tiempo para que los empleados se concentren en tareas de mayor valor añadido lo que puede traducirse en un mejor aprovechamiento del talento y mayor productividad.

# Análisis de necesidades y requerimientos

## Descripción del problema actual

Actualmente Atlantic City enfrenta desafíos significativos debido a la fragmentación de sus sistemas de gestión de clientes. La existencia de múltiples aplicaciones aisladas para manejar inscripciones, promociones, membresías y otras operaciones ha generado duplicación y dispersión de datos, lo que dificulta obtener una visión unificada de las interacciones con los clientes. Esta falta de integración se traduce en procesos manuales, errores frecuentes y una atención al cliente inconsistente, impidiendo además la implementación de estrategias basadas en información en tiempo real. Como resultado, la empresa sufre ineficiencias operativas que afectan la calidad del servicio, la personalización de las ofertas y su competitividad en el mercado del entretenimiento.

## Identificación de Stakeholders

Usuarios directos:

* *Empleados del sistema*: Son quienes interactuarán con el sistema, necesitan una interfaz sencilla y eficiente, para registrar, modificar y consultar información. Dependen de la confiabilidad del sistema para realizar sus actividades sin errores ni retrasos, requieren ciertos niveles de accesos según sus funciones, evitando que puedan modificar información crítica sin autorización.
* *Administradores o supervisores:* Tienen permisos avanzados para gestionar datos, aprobar cambios y generar reportes, además supervisan la correcta operación del sistema y garantizan que la información esté actualizada y bien estructurada. Pueden necesitar funciones adicionales como exportación de datos o monitoreo de registros de actividad

Usuario Indirectos:

* *Gerencia o directivos:*  Utilizan reportes e indicadores del sistema para evaluar rendimiento, eficiencia y tendencias. Pueden solicitar mejoras o adaptaciones en el sistema según las necesidades del negocio, y no requieren acceso completo al sistema, si no herramientas que les permitan visualizar datos procesados y analizar métricas clave.
* *Auditores o personas de control:* Se encargan de verificar la integridad y seguridad de la información almacenada, pueden enviar registros de actividad y acceso para detectar posibles inconsistencias o fallos en el proceso
* *Equipo de desarrollo y mantenimiento:* Diseñan, implementan y actualizan las funcionalidades del sistema, responden a errores o problemas técnicos que puedan surgir durante el uso del usuario. Evalúan mejoras en la eficiencia y usabilidad del sistema para optimizar la experiencia del usuario.

## Requerimientos funcionales y no funcionales

Requerimientos funcionales:

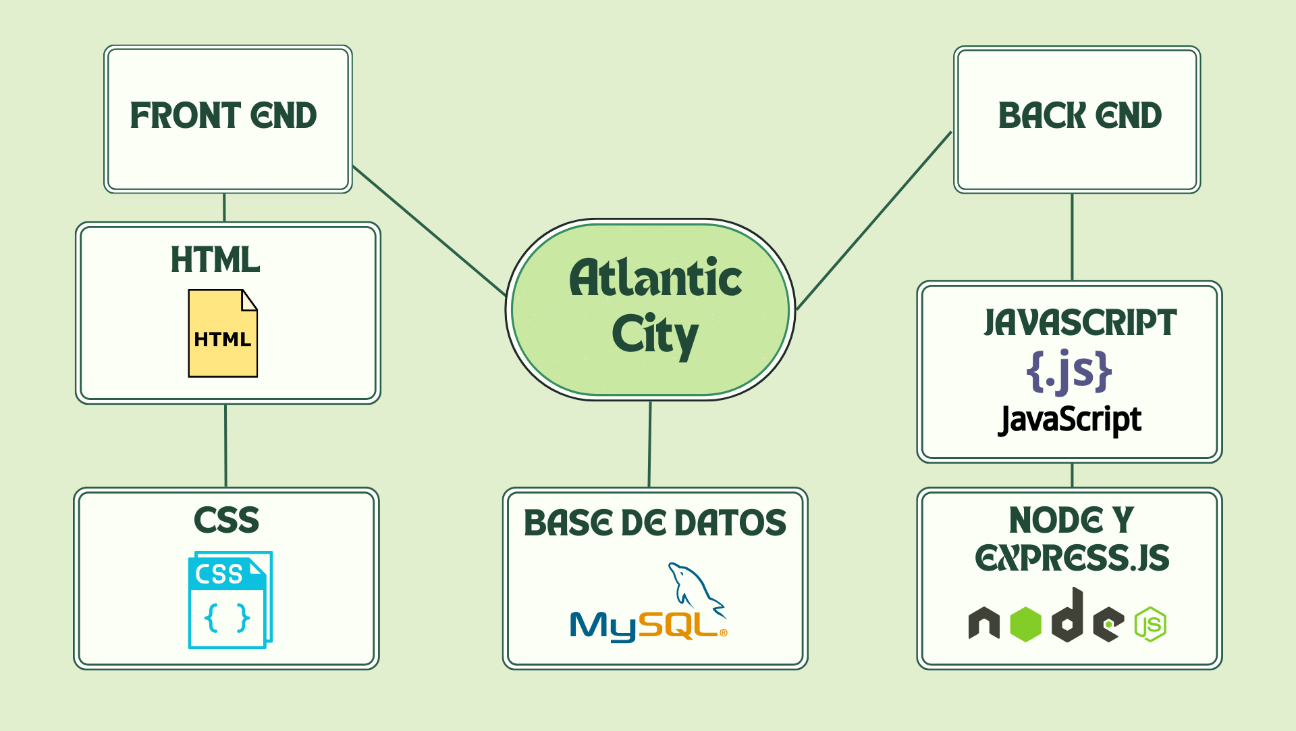
1. *Gestión centralizada en clientes* **(Alta prioridad)**: El sistema debe permitir el almacenamiento y consulta de la información de los clientes en una base de datos unificada, consolidando los datos personales, historial de visitas, e interacciones previas para mejorar la atención personalizada.
2. *Segmentación de clientes* **(Alta prioridad):** Se debe implementar una función que permita clasificar a los clientes según criterios como frecuencia de visitas, nivel de gasto y preferencias de entretenimiento (Alta prioridad): Esto facilitará la personalización de promociones y la toma de decisiones estratégicas.
3. *Automatización de procesos operativos* **(Alta prioridad)**: Se deben automatizar tareas repetitivas, como el envío de promociones y la actualización de los datos de clientes, reduciendo la carga de trabajo manual y aumentando la eficacia operativa.
4. *Gestión de promociones y ofertas personalizadas* **(Media prioridad):** El sistema debe permitir la creación, administración y asignación de promociones personalizadas en función del comportamiento del cliente, asegurando que cada usuario reciba ofertas relevantes que cultiven su lealtad y consumo.
5. *Reportes y análisis de datos* **(Baja prioridad):** Se deben generar informes visuales y detallados sobre interacciones de clientes, efectividad de promociones y tendencias de consumo. Esto permitirá tomar decisiones basadas en datos en tiempo real.

Requerimientos no funcionales:

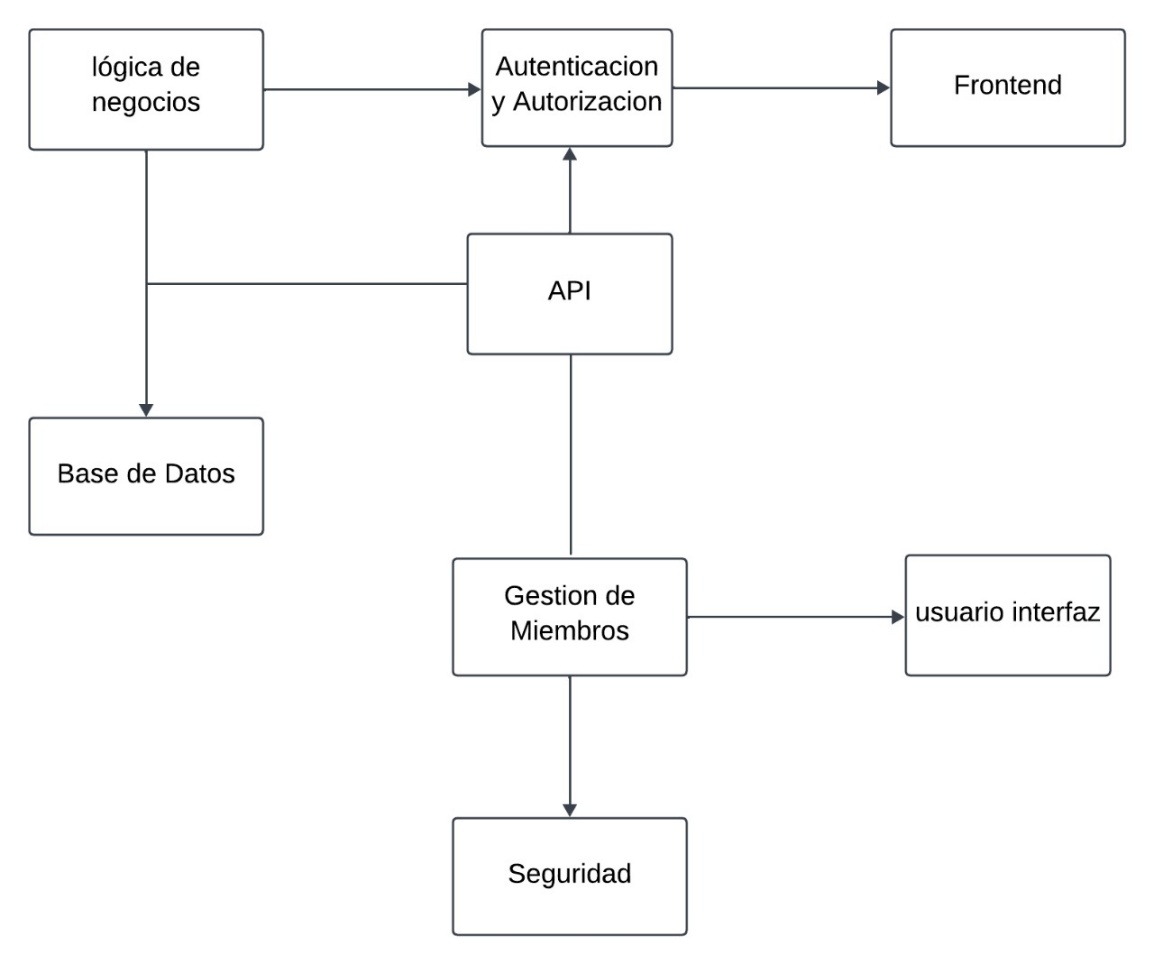
1. *Seguridad y protección de datos* **(Alta prioridad):** El sistema debe garantizar la confidencialidad e integridad de los datos personales y financieros de los clientes mediante protocolos de encriptación y acceso restringido.
2. *Escalabilidad* **(Alta prioridad):** Se debe diseñar el sistema de manera modular para que pueda crecer y adaptarse a nuevas necesidades sin comprometer su rendimiento.
3. *Integración con sistemas existentes* **(Media prioridad):** Se debe garantizar la compatibilidad e integración del nuevo sistema con los sistemas de gestión y base de datos actuales de Atlantic City, evitando duplicidad de información y optimizando procesos.
4. *Facilidad de uso e interfaz intuitiva* **(Baja prioridad):** La interfaz de usuario para el empleado debe ser clara y amigable, permitiendo una rápida adopción por parte de los empleados sin necesidad de capacitación extensa.
5. *Rendimiento, disponibilidad y confiabilidad* (**Baja prioridad)**: El sistema debe procesar grandes volúmenes de datos de manera eficientes, asegurando tiempos de respuesta óptimos en la consulta y generación de reportes, también debe garantizar una alta disponibilidad para minimizar tiempos de inactividad y evitar interrupciones en las operaciones diarias de la empresa.

# Diseño del sistema

## Arquitectura del sistema

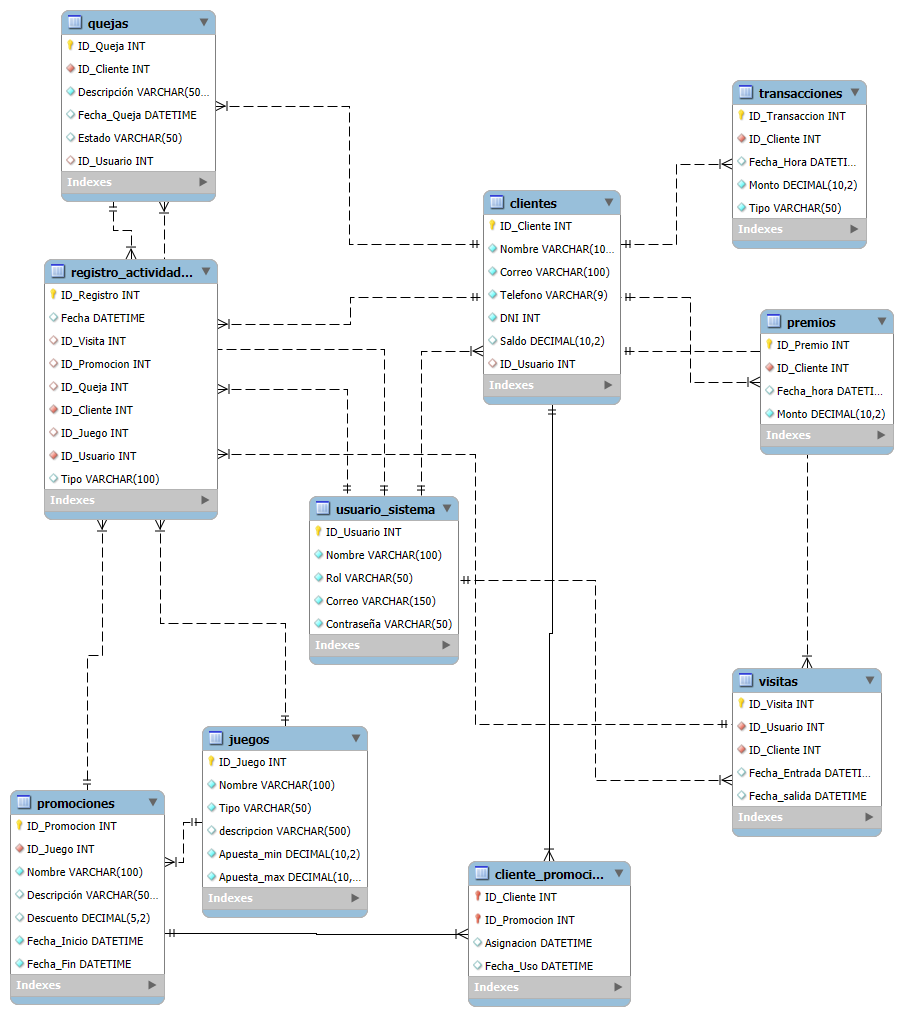


## Diagrama de componentes



## 

## Diseño de la base de datos



La base de datos ha sido diseñada para gestionar de manera eficiente las operaciones del casino. Su estructura permite el registro y control de las actividades de los clientes, las apuestas, promociones, premios, pagos y otros aspectos clave de la administración del establecimiento.

La base de datos está compuesta por diversas entidades interrelacionadas, cada una de ellas con un propósito específico:

Clientes

Tabla: Clientes

* Descripción: Almacena la información personal de los clientes del casino.

Juegos

Tabla: Juegos

* Descripción: Registra los juegos disponibles en el casino.

Apuestas

Tabla: Apuestas

* Descripción: Guarda información sobre las apuestas realizadas por los clientes.

Promociones

Tabla: Promociones

* Descripción: Contiene datos sobre promociones especiales que el casino ofrece a sus clientes.

Clientes - Promociones

Tabla: Clientes\_Promoción

* Descripción: Relaciona clientes con promociones aplicadas.

Registro de Actividades

Tabla: Registro\_Actividades

* Descripción: Almacena un historial de eventos dentro del casino.

Premios

Tabla: Premios

* Descripción: Registra los premios otorgados a los clientes.

Transacciones

Tabla: Transacciones

* Descripción: Lleva el control de las transacciones financieras dentro del casino.

Visitas

Tabla: Visitas

* Descripción: Registra cada visita de un cliente al casino.

Quejas

Tabla: Queja

* Descripción: Registra cada queja de un cliente al casino.

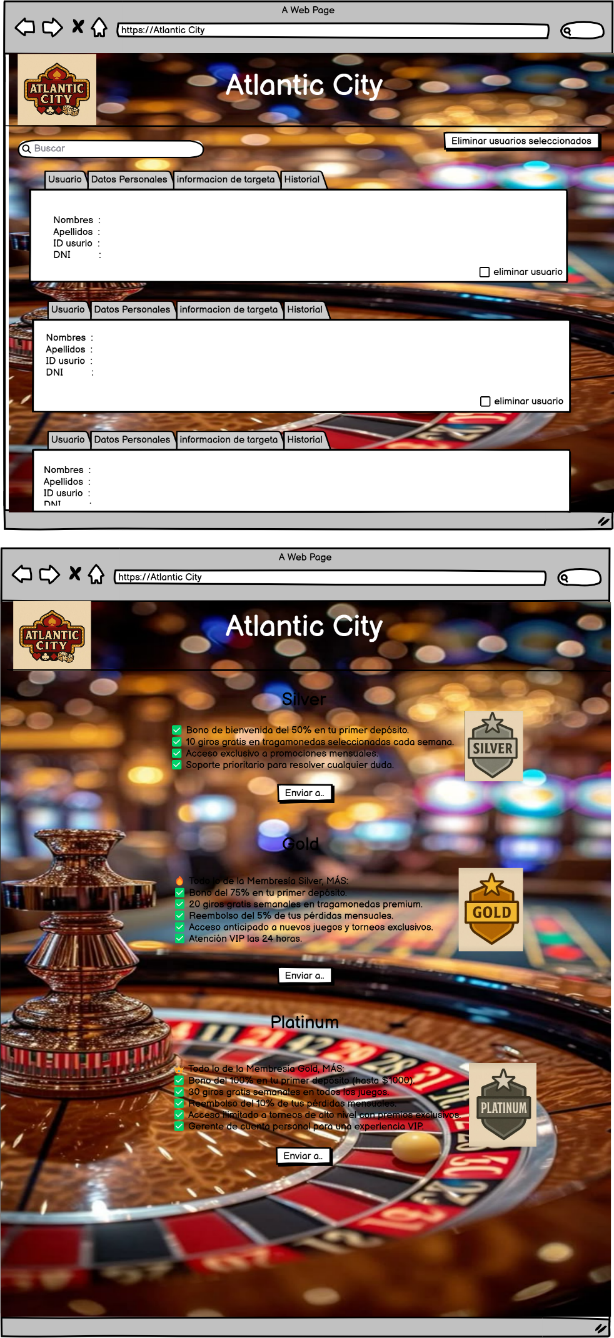
Usuarios

Tabla: Usuario\_Sistema

* Descripción: Entidad encargada de registrar las actividades del casino

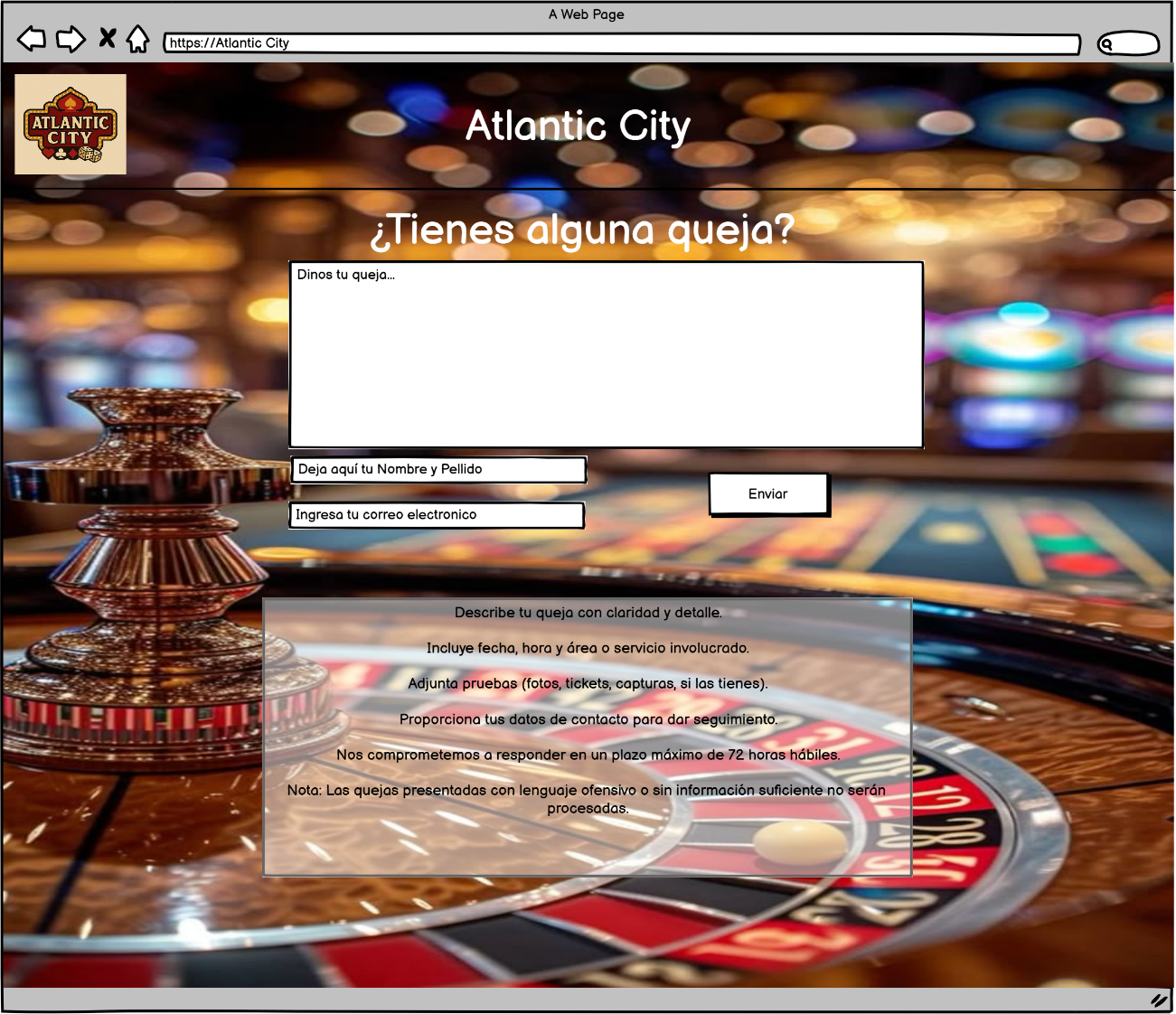
## Prototipo de interfaz de usuario

Página de inicio



Página de reportes

Página de Quejas



## Flujo de usuario y funcionalidades

Inicio

Representa el ingreso del cliente ala aplicación.

Registro / Identificación (si aplica)

En algunos casinos se requiere verificar la edad o registrarse si es socio.

Encuentra su juego

**Selección del juego**

Puede elegir entre:

* + Máquinas tragamonedas
  + Mesas (póker, ruleta, blackjack, etc.)
  + Apuestas deportivas (si el casino las tiene)

Hace su Apuesta

El cliente cambia dinero por crédito o carga crédito en una tarjeta de juego.

Ciclo de juego

Aquí el jugador:

* + Apuesta
  + Gana o pierde
  + Decide si sigue o se retira

Este ciclo puede repetirse varias veces hasta que decida parar.

Cobro de ganancias

Si el jugador tiene fichas o crédito acumulado, puede cambiarlas por dinero.

Retiro del casino

El jugador sale del casino.

Fin

Diagrama

Descripción generada automáticamente

# 

Diagrama de flujo 2 Atlantic City

Explicación Diagrama de flujo:

1. Inicio El usuario se registra o inicia sesión en la plataforma
2. Verificación de identidad / ingreso
3. Cambio de dinero por saldo. Si no tiene saldo, debe hacer un depósito a su cuenta
4. Selección de actividad (juego)
   * Tragamonedas
   * Ruleta
   * Póker
   * Blackjack
5. Selecciona el juego y realiza su apuesta
6. La plataforma procesa la apuesta y la guarda
7. Según el resultado:
   1. Si gana, acreditan las ganancias en su cuenta
   2. Si pierde, finaliza la apuesta
8. Decisión: seguir jugando o no
9. El usuario puede optar por retirar fondos o guardarlos en su cuenta
10. Salida de la aplicación
11. Fin

# Desarrollo y funcionalidad

## Visual Studio Code

En el desarrollo del sistema de información de gestión de clientes para Atlantic City, se ha optado por Visual Studio Code como el entorno principal de programación. Esta herramienta representa una opción eficiente para proyectos que requieren integrar múltiples tecnologías, ya que se adapta fácilmente a diferentes lenguajes de programación y entornos de ejecución. Su interfaz es limpia, ligera y permite una organización intuitiva de los archivos del proyecto, lo que favorece un desarrollo más ordenado y enfocado.

Uno de los motivos fundamentales de su elección en su amplio catálogo de extensiones, que facilita la integración de funcionalidades específicas como el manejo de base de datos, la conexión con servidores y el control de versiones. Estas características permiten trabajar con mayor precisión y autonomía, optimizando el tiempo y reduciendo errores.

Asimismo, su compatibilidad con herramientas como Node.js, Express, Java y el uso de lenguajes para el frontend como HTML y CSS lo convierten en una plataforma versátil para todo el ciclo de desarrollo. Todo ello hace que Visual Studio Code sea una elección estratégica que potencia la productividad del equipo y mejora la calidad del producto final.

## Base de datos MySQL Workbench

Para la gestión de información de base de datos en Atlantic City, se ha seleccionado MySQL Workbench como herramienta de gestión y diseño de la base de datos. Esta selección responde a la necesidad de contar con una plataforma confiable que permita estructurar, visualizar y mantener la integridad de los datos de manera eficiente y segura.

Una de las principales razones de su uso es su capacidad para modelar esquemas relacionales de forma gráfica, lo cual facilita la organización lógica de los datos y la comprensión del modelo. Esto es especialmente valioso en un proyecto donde se requiere que la base de datos este bien estructurada, minimice redundancias y asegure la coherencia de la información registrada.

Además, MySQL Workbench permite realizar tareas administrativas como la gestión de usuarios, la ejecución de consultas SQL y la supervisión del rendimiento. Esta funcionalidad integral reduce la dependencia de múltiples herramientas y centraliza el control sobre la base de datos en una sola aplicación.

Finalmente, su compatibilidad con motores de base de datos ampliamente utilizados y su facilidad de integración con tecnologías como Node.js y Java lo convierten en una solución ideal para proyectos modernos y escalables, como el que estamos desarrollando para Atlantic City.

## Node.js

Para el desarrollo del backend del proyecto, se optó por utilizar Node.js como entorno de ejecución JavaScript del lado del servidor. Esta elección responde a la necesidad de contar con una plataforma moderna, rápida y flexible que permite manejar múltiples operaciones y conexiones con eficiencia.

Permite gestiona múltiples solicitudes simultáneamente sin afectar el rendimiento del sistema gracias a su modelo de ejecución asincrónico y no bloqueantes, además Node permite el uso de JavaScript tanto en el frontend como en el backend, lo que facilita el desarrollo y mantenimiento del código al utilizar un único lenguaje en todo el proyecto. Este enfoque unificado mejora la comunicación entre distintas partes del sistema y reduce la complejidad del desarrollo.

Finalmente, la integración de Node.js con herramientas como MySQL Workbench para la gestión de datos, y su estructura modular, lo convierten en una tecnología ideal para desarrollar proyectos escalables, organizados y con buen rendimiento.

## Implementación del Frontend

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

El Frontend del proyecto esta guardado en la carpeta “Public” que cuenta con los archivos:

.HTML

Cada uno de estos archivos define una página web estática con estructura semántica y sirve como punto de entrada para el usuario, se eligió el html5 porque cuenta con una estructura semántica que mejora la accesibilidad y describe mejor el contenido para que los motores de búsqueda entiendan mejor la página, además soporta formularios, multimedia e integra con facilidad tecnologías modernas del lado del servidor como Node.js y Java.

En nuestro proyecto se requería que el Frontend se comunique correctamente con el Backend que está hecho con Node.js y esto es posible gracias a que HTML5 puede enviar datos con métodos POST y GET además entiende bien las funciones de fetch() desde el main.js para hacer una solicitud a las rutas del backend.

En el proyecto contamos con los archivos:

* Clientes.html: Interfaz para ver/gestionar clientes
* Inicio.html: Es la página de bienvenida o principal
* Juegos.html: La página que contiene el listado y detalles de cada juego disponible
* Login.html: Es el formulario de autenticación para los clientes y empleados
* Promociones.html: Muestra las promociones vigentes que están disponible para cada cliente
* Quejascliente.html: Muestra el formulario donde el cliente puede realizar alguna queja
* Transacciones.html: Es el historial o gestor de transacciones.

.CSS

En este archivo se define la apariencia de la página (colores, tipografías, márgenes, disposición con Flexbox/Grid responsive, etc) y se enlaza a cada html mediante:

<link rel=”stylesheet” hreef=”estilos.css”>

Main.js

Aquí se gestiona la interactividad de los usuarios con la página (clicks, envíos de formularios, manipula el DOM cambiando el contenido de la página de forma dinámica. Se conecta con el API al Backend usando fetch() para leer o enviar datos (por ejemplo al obtener la lista de clientes o enviar una nueva queja).

Se incluye en cada HTML con:  
<script src=”main.js”></script>

## Implementación del Backend

Imagen que contiene Texto

Descripción generada automáticamente

Para la programación del Backend se están utilizando las tecnologías:

JavaScript

Es un lenguaje de programación interpretado que hoy en día junto a Node.js se usa en el servidor. Al utilizar JavaScript tanto en el frontend como en el backend se unifica el lenguaje de desarrollo, facilitando el mantenimiento, reduciendo errores de compatibilidad.

A través de la implementación de APIs REST, que son independientes del lenguaje. Java puede consumir o exponer servicios compatibles con las rutas que ya existen en Node.js, logrando así una comunicación entre sistemas diferentes mediante solicitudes HTTP.

Se usa en app.js, db.js y los archivos dentro de rutas/ y contienen toda la lógica del servidor escrita en JavaScript.

Node.js

Es un entorno de ejecución para JavaScript del lado del servidor. Se eligió porque permite trabajar en JavaScript tanto en el cliente como en el servidor, lo que unifica el lenguaje y simplifica el desarrollo. Además, tiene la capacidad de manejar múltiples conexiones simultáneas lo que lo hace ideal para el proyecto al contar con muchas interacciones

Se utiliza dentro de:

* **App.js** que actúa como punto de entrada del servidor, inicializa la aplicación, importa las rutas y configura el puerto en el que el servidor este.
* En **los archivos de la carpeta rutas/** que contienen las API REST encargadas de recibir las solicitudes del frontend y responder con datos desde la base de datos.
* En **db.js** para establecer y exportar la conexión a la base de datos MySQL, utilizando módulos compatibles con Node.js con mysql2

MySQL

Es un sistema gestión de base de datos relacionales y nos permite almacenar, organizar y consultar los datos mediante lenguaje SQL.

MySQL es eficiente, ampliamente soportado, gratuito y confiable para trabajar con grandes volúmenes de datos estructurados. Su integración con Node.js es fluida mediante librerías como mysql2

Se utiliza dentro del archivo:

basedatos/db.js en donde se configura la conexión a la base de datos indicando el host, el usuario, la contraseña y el nombre de la base.

Cada archivo de rutas/ utiliza esta conexión para realizar consultas SQL (SELECT, INSERT, UPDATE, DELETE) según la funcionalidad requerida

Descripción de archivos del backend

App.js: Punto de entrada del servidor, donde se configuran las rutas y puertos

Basededatos/db.js: Conexión a la base de datos MySQL, define los parámetros y se exporta.

Rutas/clientes.js: Ruta relacionada a operaciones sobre clientes (registro, consulta, etc)

Rutas/juegos .js: Ruta para gestionar juegos disponibles

Rutas/promociones.js: Ruta para gestionar promociones o descuentos

Rutas/quejas.js: Ruta para el registro y visualización de quejas de clientes.

Rutas/transacciones.js: Ruta para operaciones financieras o registro de transacciones.

Rutas/package.json y lock.json: Archivos de configuración de Node.js que contienen las dependencias del proyecto y asegura versiones exactas de dependencias

## Conexión a la base de datos

Texto

Descripción generada automáticamente

Este archivo tiene como propósito general establecer la conexión entre la aplicación Node.js y la base de datos MySQL. Es un módulo que exporta una conexión activa para que otros archivos del backend puedan hacer consultas SQL a la base de datos de Atlantic\_City

Texto

Descripción generada automáticamenteEjemplo en Clientes.js:

Se importa el módulo MySQL, que es una biblioteca de Node.js que permite conectarse a bases de datos MySQL, ejecutar consultas y gestionar conexiones.

Luego se configura la conexión con los siguientes parámetros:

* La dirección del servidor de base de datos en este caso “127.0.0.1” que indica el localhost
* El usuario de la base de datos “root” que es el predeterminado
* La contraseña de acceso a MySQl
* Y el nombre de la base de datos a la que se conecta (Atlantic\_City)

Se intenta conectar y se imprime un mensaje en la consola según el resultado:

* Si ocurre un error, lo muestra en console.log
* Si la conexión es exitosa confirma que la base de datos esta lista para usarse

Por último, la exportación del modulo que permite que la variable conexión este disponible para ser usada en otros archivos del proyecto: Por ejemplo, en app.js o los archivos dentro de rutas/ pueden importar esta conexión para ejecutar consultas SQL.

## Estructura de archivos

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Basededatos/

Esta carpeta contiene el archivo encargado de establecer la conexión con la base de datos MySQL. En este caso, el archivo principal es db.js

Node\_modules/

Contiene todas las dependencias que se instalaron con Node.js usando npm. No se modifica directamente

Public/

Contiene todos los archivos del frontend accesibles por el navegador del usuario. Aquí se incluyen los archivos .html, los archivos .css, y el archivo main.js

Rutas/

Contiene los archivos que definen las rutas de la API REST del backend. Cada archivo maneja un conjunto de funciones relacionadas con un módulo del sistema, aquí se contienen los crud de las páginas de public.

# Presentación final del sistema

## Reflexión sobre el proceso de desarrollo

El desarrollo de este proyecto impulso el proceso creativo, lógico y colaborativo que refleja la capacidad de transformar ideas en soluciones reales. A lo largo del proceso de desarrollo desde el análisis de requerimientos hasta la implementación y prueba del sistema se ponen en práctica diversas habilidades fundamentales que son parte de nuestra carrera.

Participar en un proyecto de desarrollo, como puede ser la construcción de un sistema para un casino digital, permite aplicar conocimientos técnicos en un entorno práctico. Esto incluye desde la lógica de programación y diseño de bases de datos, hasta conceptos más complejos como la seguridad, usabilidad, integración de APIs y arquitecturas escalables. Pero también enseña a trabajar bajo metodologías ágiles, colaborar en equipo, documentar adecuadamente y pensar en la experiencia del usuario.

Además, enfrentar los desafíos del desarrollo errores inesperados, cambios en los requisitos, optimización del rendimiento fomenta la resolución de problemas, el pensamiento crítico y la adaptabilidad, habilidades altamente valoradas en el mundo laboral.

Cada etapa del desarrollo fortalece la formación profesional y personal. Se aprende no solo a construir software, sino a construir soluciones que impactan a los usuarios reales. Este proceso nos ayuda a tener la capacidad de adaptarnos al ritmo de la industria tecnológica y aportar valor en múltiples contextos.

## Conclusiones y recomendaciones

Logramos un avance significativo en el backend ya que se logro implementar una estructura funcional del backend utilizando node.js como entorno de ejecución y MySQL como sistema de gestión de base de datos. Esto incluye la conexión a la base de datos, la creación de rutas para cada entidad y la integración de estas con app.js

Además, el proyecto está organizado en carpetas para separar de forma clara el backend, pero, aunque se crearon varias páginas html y se estableció la base para los estilos, no se completó la programación de todas las interfaces necesarias para que el usuario final pueda interactuar con el sistema de manera completa, además se planteo que el proyecto fuera compatible con tecnologías como Java y Node.js sin embargo, la parte en Java no se terminó de implementar.

Recomendaciones

Es importante terminar el desarrollo de las interfaces html, conectarlas con el backend a través de formularios o llamadas. Aún faltan implementar la lógica completa en los módulos del backend y vincular las rutas con acciones concretas en la base de datos, continuar el desarrollo también en Java, se debe definir claramente el rol de cada tecnología en el sistema y asegurar que puedan comunicarse de manera adecuada.

# Anexos

## Código fuente

<https://github.com/memo398/Final-Atlantic-City>

## Materiales visuales

## App.js

Texto

Descripción generada automáticamente

## Db.js

Texto

Descripción generada automáticamente

## Ejemplo de CRUD en clientes.js

Texto

Descripción generada automáticamente

Captura de pantalla de computadora

Descripción generada automáticamente