**Mejoras a sistema de caja FastPos para pizzerías**

**Informe Proyecto APT**

**Fernanda Monsalve Luco - Matías Gutierrez**

**CAPSTONE 002D**

[**Relación con el perfil de egreso 3**](#_heading=h.q7vss644qlnw)

[**Relación con intereses profesionales 4**](#_heading=h.lbamxaszpzv0)

[**Factibilidad del proyecto 4**](#_heading=h.cs5njvvxz9au)

[**Cumplimiento de indicadores de calidad 5**](#_heading=h.itixqjj2znsn)

[**Abstract, conclusiones y reflexión (en inglés) 6**](#_heading=h.pszk692nggok)

[Plan de Pruebas – Cálculo de Pedido con Ingredientes Adicionales 9](#_heading=h.diz9cc9d0fwb)

[Módulo: Gestión de Pedidos / Caja 9](#_heading=h.k0rbigvqxmmb)

[Responsable: Equipo de QA 9](#_heading=h.b5a1dde9g8wk)

[Fecha: [15/08/2025] 9](#_heading=h.k0rbigvqxmmb)

[Tabla de Casos de Prueba 9](#_heading=h.rd1oled9cd6x)

[Prueba de Integración – Gestión de Pedido y Actualización de Inventario 10](#_heading=h.9hknnf1qom0l)

[Escenario de prueba 10](#_heading=h.fnxspqcp4i3o)

[Pasos de prueba 11](#_heading=h.ghd8rlfphoz3)

[Resultados esperados 11](#_heading=h.l1fyq3d1ni2k)

[Notas adicionales 12](#_heading=h.uc3ce5zi68cg)

[Prueba de corrección – Eliminación de pedido e inventario 12](#_heading=h.1w0t2uirassq)

[Escenario inicial 12](#_heading=h.tmfk8h7i9htm)

[Pasos de prueba 13](#_heading=h.pz0k389zvxw6)

[Resultados esperados tras corrección 13](#_heading=h.22bk60snasdc)

[Notas 14](#_heading=h.ykvt2enet238)

[Cronograma de Desarrollo – Proyecto FastPOS 14](#_heading=h.3q3z3wrrgpq2)

[Cronograma 15](#_heading=h.tgb620olmrhj)

[Planificación del Desarrollo – Cronograma de Módulos 15](#_heading=h.n90h2s8t1k7g)

[Notas adicionales 16](#_heading=h.xm5tdwulwz3z)

[Gestión de Incidencias – Corrección de Cálculo de Impuestos 16](#_heading=h.qte0ko3xamfl)

[Diseño de la colección Pedidos en Firestore 17](#_heading=h.wghyvn2u00ra)

[Implementación de colecciones y reglas de seguridad en Firestore 18](#_heading=h.otvyd7c81hvo)

[Colecciones implementadas 19](#_heading=h.egiep84xsp14)

[Reglas de seguridad aplicadas 19](#_heading=h.693t3cjdhz5h)

[**Resultados esperados 20**](#_heading=h.5hch3dpye4zo)

[Desarrollo del Frontend en React 20](#_heading=h.ttplcgl755ra)

[Componentes principales 20](#_heading=h.sqpiwfab7q0)

[Buenas prácticas aplicadas 21](#_heading=h.7i0uj9gmn7iq)

[Resultado esperado 21](#_heading=h.n18eeu56vfhi)

[Integración de módulos – Pedido, Inventario y Caja 22](#_heading=h.4os44u1g8ci)

[Flujo de integración 22](#_heading=h.e7k6vkcxmome)

[Ejemplo práctico 23](#_heading=h.vk2hx3gh8onu)

[Notas 23](#_heading=h.fvnjwprv6yo2)

[Competencia asociada: 23](#_heading=h.un9r02b3kb4b)

[Tabla Resumen de Evidencias por Rol – FastPOS 23](#_heading=h.g9ruoe1zzhkr)

[Notas 25](#_heading=h.2tmwoxrmti44)

Descripción del Proyecto

El Proyecto APT consiste en el desarrollo de un sistema FastPOS para pizzerías, cuyo propósito es optimizar la gestión de pedidos, el cálculo automático de precios e impuestos, los procesos de pago y la administración de inventario.

**Alcance contemplado:**

* Módulo de autenticación con roles diferenciados (mesero, caja y administrador).
* Gestión de pedidos desde la toma en mesa hasta el pago en caja.
* Administración de inventario, productos y precios.
* Generación de reportes básicos de ventas.

**Fuera de alcance:**

* Integración con plataformas de delivery externas.
* Procesamiento de pagos electrónicos reales (se trabajará con simulación).
* Implementación de inteligencia de negocio avanzada.

De esta forma, el proyecto presenta límites claros, alcanzables y pertinentes al marco de la asignatura, asegurando que los entregables sean coherentes con los recursos y tiempos disponibles.

## Relación con el perfil de egreso

El proyecto se vincula directamente con las competencias del perfil de egreso de Ingeniería en Informática, y en cada una se detalla cómo se aplica:

* **Desarrollar soluciones de software utilizando lenguajes de programación modernos:** se implementa el frontend en **React** y el backend en **Node.js**, aplicando buenas prácticas de programación.
* **Diseñar y administrar bases de datos relacionales:** se desarrolla un modelo en **SQL Developer** para gestionar pedidos, inventario y productos.
* **Aplicar metodologías de desarrollo de software:** se emplea **Scrum**, con sprints semanales, backlog definido y reuniones de control de avance.
* **Diseñar interfaces de usuario aplicando principios de usabilidad:** se prototipa pantallas en **Figma**, priorizando la experiencia del usuario final.
* **Trabajar en equipo y comunicar resultados:** se practica mediante documentación, reportes de avance y la presentación final del proyecto.

Estas competencias no solo se mencionan, sino que se explican en relación con tareas concretas, asegurando la aplicación directa del perfil de egreso al desarrollo del sistema.

## Relación con intereses profesionales

Mis intereses profesionales se enfocan en el **desarrollo de sistemas empresariales** que permitan **automatizar procesos y mejorar la eficiencia de los negocios**. El Proyecto APT refleja este interés, ya que integra tecnologías modernas en un contexto realista, reforzando mi perfil hacia el **desarrollo de soluciones informáticas de gestión**. Esta experiencia práctica me prepara para enfrentar proyectos similares en entornos laborales reales.

## Factibilidad del proyecto

El proyecto es factible de realizar en el marco de la asignatura debido a los siguientes elementos:

* **Definición clara de alcance:** los módulos están acotados (login, caja, mesero, administración), permitiendo entregables progresivos y medibles.
* **Planificación inicial conocida:** se organiza en un cronograma de **7 semanas** (sprints), donde cada semana se entrega un módulo funcional.
* **Gestión de riesgos:** se consideran posibles problemas como retrasos, errores de integración o ajustes en diseño. Como mitigación se aplicarán **pruebas unitarias**, priorización de funcionalidades críticas y reuniones semanales de control.
* **Metodología elegida:** se utiliza **Scrum**, que facilita el trabajo incremental, la entrega continua de valor y la adaptación ante imprevistos.
* **Recursos disponibles:** tecnologías accesibles y conocidas (React, Node.js, SQL Developer, Figma).

En conjunto, estos factores fortalecen la viabilidad del proyecto dentro del tiempo y marco académico establecido.

## Cumplimiento de indicadores de calidad

El diseño del Proyecto APT responde a los **indicadores de calidad definidos por la disciplina**:

* **Claridad y orden en la presentación:** se entrega informe estructurado con objetivos, descripción, metodología, planificación y evidencias.
* **Uso de lenguaje disciplinar apropiado:** se utilizan términos técnicos como **POS, frontend, backend, base de datos relacional, UI/UX, arquitectura cliente-servidor**.
* **Consistencia con la carrera:** integra desarrollo de software, modelado de bases de datos y experiencia de usuario.
* **Metodología validada:** aplicación de **Scrum**, reconocida en la industria como estándar de gestión de proyectos de software.
* **Propuesta técnica fundamentada:** definición clara de roles de usuario, flujo de procesos y modularidad del sistema.
* **Evidencias contempladas:** prototipos en Figma, código en GitHub, pruebas funcionales, retroalimentación de usuarios y demo final.

De esta manera, el proyecto cumple con los estándares de calidad esperados en Ingeniería en Informática.

## Abstract, conclusiones y reflexión (en inglés)

**Abstract**

The APT project aims to develop a FastPOS system for pizzerias, designed to improve order management, payment processes, and inventory control. The system integrates frontend and backend technologies, providing a user-friendly interface for staff and reliable database management. This project demonstrates the practical application of informatics engineering skills such as programming, database design, and UI/UX development. Its relevance lies in its potential to enhance business efficiency and customer service while consolidating professional competencies.

**Conclusions**

The APT project highlights the applicability of academic knowledge to real-world business environments. It shows the importance of integrating programming, database design, and user interface development to create practical solutions for organizations. The use of agile methodologies, particularly Scrum, contributed to the feasibility of the project by ensuring incremental progress and adaptability. Overall, this project reinforced my professional interest in business management systems and strengthened my teamwork, planning, and problem-solving skills.

**Reflection**

Developing this project allowed me to connect theory with practice, turning classroom learning into a functional product. I realized the importance of balancing technical performance with usability, ensuring that the system is efficient and intuitive. Through agile development, I learned that continuous iteration and feedback produce stable results. This experience motivates me to keep specializing in software engineering, with a focus on business solutions that improve efficiency and decision-making.

| **Competencia** | **Indicador de Calidad** | **Ejemplo Concreto de Evidencia en FastPOS** |
| --- | --- | --- |
| Pruebas de certificación de productos y procesos | Diseña pruebas de validación | Se creó un plan de pruebas donde se validaba que al agregar un pedido de pizza con ingredientes adicionales, el precio total se calculaba correctamente según impuestos y descuentos. |
|  | Aplica pruebas de validación | Se realizaron pruebas de integración: un mesero agregaba pedidos, el sistema actualiza inventario automáticamente y calculaba totales en caja. |
|  | Desarrolla mejoras | Tras detectar que el sistema no actualizaba correctamente el inventario al eliminar un pedido, se corrigió la función de eliminación y se volvió a probar con éxito. |
| Gestión de proyectos informáticos | Planifica proyectos | Se elaboró un cronograma de desarrollo asignando tiempos para cada módulo: autenticación, gestión de pedidos, inventario y pagos. |
|  | Controla proyectos | Al detectarse un error en el cálculo de impuestos, se asignó la tarea de corrección al equipo de frontend sin retrasar la entrega final. |
| Construcción de modelos de datos | Diseña modelos de datos | Se diseñó la colección Pedidos en Firestore con campos como idPedido, idCliente, items, precioTotal, estado y fecha, considerando futuras consultas por fecha y estado. |
|  | Implementa modelos de datos | Se implementaron las colecciones Usuarios, Roles, Productos, Pedidos con reglas de seguridad que restringen el acceso a usuarios autenticados. |
| Desarrollo de soluciones de software | Construye soluciones | Se desarrolló el frontend en React, con componentes separados para Login, Mesa, Pedido e Inventario, siguiendo patrones de diseño reutilizables. |
|  | Integra componentes | Se conectaron los módulos: al seleccionar productos en Pedido, Inventario se actualizaba automáticamente y los totales se reflejaban en Caja. |
|  | Implanta la solución | Se desplegó la app en Firebase Hosting (https://fastpos-pizzeria.web.app) y se verificó que los distintos roles pudieran usar todas las funcionalidades sin errores. |

## Plan de Pruebas – Cálculo de Pedido con Ingredientes Adicionales

Objetivo: Validar que el sistema calcule correctamente el precio total de un pedido de pizza considerando ingredientes adicionales, impuestos y descuentos aplicables.

### Módulo: Gestión de Pedidos / Caja

### Responsable: Equipo de QA

### Fecha: [15/08/2025]

### Tabla de Casos de Prueba

| **ID** | **Caso de Prueba** | **Datos de Entrada** | **Pasos** | **Resultado Esperado** | **Estado** |
| --- | --- | --- | --- | --- | --- |
| **TP-001** | Pedido sin ingredientes adicionales | Pizza Mediana ($5000), sin extra | 1. Seleccionar Pizza Mediana.2. No agregar ingredientes adicionales.3. Confirmar pedido. | Precio subtotal = $5000IVA 19% = $950Total = $5950 | Pendiente |
| **TP-002** | Pedido con 2 ingredientes adicionales | Pizza Grande ($7000), extra pepperoni ($1000) y extra champiñones ($500) | 1. Seleccionar Pizza Grande.2. Agregar pepperoni (+$1000) y champiñones (+$500).3. Confirmar pedido. | Precio subtotal = $7000 + $1000 + $500 = $8500IVA 19% = $1615Total = $10115 | Pendiente |
| **TP-003** | Pedido con descuento por promoción | Pizza Familiar ($10000), sin extra, 10% descuento | 1. Seleccionar Pizza Familiar.2. Aplicar descuento promocional 10% .3. Confirmar pedido. | Precio subtotal = $10000 - 10% = $9000IVA 19% = $1710Total = $10710 | Pendiente |
| **TP-004** | Pedido con ingredientes extra y descuento | Pizza Mediana ($5000), extra jamón ($800), promoción 5% | 1. Seleccionar Pizza Mediana.2. Agregar jamón (+$800).3. Aplicar descuento 5% .4. Confirmar pedido. | Precio subtotal = ($5000 + $800) \* 0.95 = $5510IVA 19% = $1046.90Total = $6556.90 | Pendiente |
| **TP-005** | Pedido con múltiples ingredientes adicionales | Pizza Grande ($7000), extra queso ($500), extra pepperoni ($1000), extra champiñones ($500) | 1. Seleccionar Pizza Grande.2. Agregar todos los extras.3. Confirmar pedido. | Precio subtotal = 7000 + 500 + 1000 + 500 = $9000IVA 19% = $1710Total = $10710 | Pendiente |

¡Perfecto Fer! Te voy a mostrar un **ejemplo detallado de prueba de integración** para tu FastPOS, basada en lo que mencionas: que un mesero agregue pedidos y el sistema actualiza inventario y calcular totales en caja. Esto simula cómo se probaría en la práctica y sirve como evidencia concreta.

## Prueba de Integración – Gestión de Pedido y Actualización de Inventario

**Objetivo:** Validar que al agregar un pedido desde el módulo Mesero, el sistema actualice automáticamente el inventario y calcule el total en Caja.

**Módulo involucrado:** Mesero, Inventario, Caja

**Responsable:** Equipo de QA

**Fecha:** [20/08/2025]

### Escenario de prueba

**Datos iniciales:**

* Inventario:
  + Pizza Mediana: 10 unidades
  + Pepperoni: 5 unidades
  + Champiñones: 8 unidades
* Precios:
  + Pizza Mediana: $5000
  + Pepperoni extra: $1000
  + Champiñones extra: $500
* IVA: 19%
* Descuento: ninguno

### Pasos de prueba

1. Mesero inicia sesión en el módulo de pedidos.
2. Selecciona **Pizza Mediana** para un cliente.
3. Agrega **pepperoni extra** y **champiñones extra**.
4. Confirma el pedido.
5. El sistema registra el pedido en la base de datos (Pedidos) y calcula el **subtotal, IVA y total**.
6. El sistema actualiza automáticamente el inventario:
   * Pizza Mediana: 10 → 9
   * Pepperoni: 5 → 4
   * Champiñones: 8 → 7
7. El total del pedido se refleja correctamente en **Caja**.

### Resultados esperados

| Concepto | Valor esperado |
| --- | --- |
| Subtotal | 5000 + 1000 + 500 = 6500 |
| IVA 19% | 6500 × 0.19 = 1235 |
| Total | 6500 + 1235 = 7735 |
| Inventario Pizza Mediana | 9 |
| Inventario Pepperoni | 4 |
| Inventario Champiñones | 7 |
| Registro en base de datos | Pedido creado con todos los detalles correctamente |

### Notas adicionales

* La prueba válida **integración entre módulos**, no solo la funcionalidad aislada del pedido.
* Permite identificar errores como:
  + Totales incorrectos por sumatoria de ingredientes.
  + Inventario que no se descuenta al confirmar un pedido.
  + Fallos de comunicación entre Mesero y Caja.

## Prueba de corrección – Eliminación de pedido e inventario

**Objetivo:** Verificar que al eliminar un pedido, el sistema actualice correctamente las cantidades en inventario.

**Módulos involucrados:** Mesero, Inventario, Caja

**Responsable:** Equipo de QA

**Fecha:** [20/08/2025]

### Escenario inicial

**Inventario antes del pedido:**

| Producto | Cantidad |
| --- | --- |
| Pizza Mediana | 10 |
| Pepperoni | 5 |
| Champiñones | 8 |

**Pedido a eliminar:**

* Pizza Mediana: 1
* Ingredientes adicionales: Pepperoni (1), Champiñones (1)

### Pasos de prueba

1. Se crea un pedido desde el módulo Mesero con los productos mencionados.
2. Se confirma el pedido, el sistema descuenta del inventario:
   * Pizza Mediana: 10 → 9
   * Pepperoni: 5 → 4
   * Champiñones: 8 → 7
3. Se detecta que al **eliminar el pedido**, el inventario **no se actualizaba correctamente** (los productos no volvían a sumar las unidades eliminadas).
4. Se corrige la función de eliminación de pedidos en el backend, asegurando que al borrar un pedido:
   * Se re-incrementen las cantidades de los productos correspondientes en inventario.
5. Se vuelve a ejecutar la prueba: eliminar el mismo pedido.

### Resultados esperados tras corrección

| Producto | Cantidad después de eliminar el pedido |
| --- | --- |
| Pizza Mediana | 10 |
| Pepperoni | 5 |
| Champiñones | 8 |

* El pedido eliminado ya no aparece en la base de datos (Pedidos).
* Los totales en Caja se actualizan correctamente.
* No se generan inconsistencias en el inventario ni en los reportes de ventas.

### Notas

* Esta prueba demuestra la mejora **continua del producto** basada en resultados de pruebas.
* Permite validar la integridad **de datos** y el correcto funcionamiento del flujo de negocio.
* Sirve como evidencia de la **competencia “Desarrolla mejoras al producto en base a resultados de pruebas”**.

## Cronograma de Desarrollo – Proyecto FastPOS

**Duración total estimada:** 6 semanas

| Módulo / Actividad | Tareas principales | Duración estimada | Semana |
| --- | --- | --- | --- |
| Autenticación y Roles | - Configuración de Firebase Auth- Creación de roles (mesero, caja, admin)- Diseño de vistas de login y control de acceso | 1 semana | 1 |
| Gestión de Pedidos | - Interfaz de toma de pedidos- Registro en Firestore- Cálculo automático de precios e impuestos- Asociación con mesa/cliente | 2 semanas | 2 – 3 |
| Inventario | - Modelo de datos de productos e insumos- Actualización automática según pedidos- Vistas de administración de stock | 1 semana | 4 |
| Pagos y Caja | - Flujo de cobro- Integración con pedidos- Actualización de caja- Reportes básicos de ventas | 1 semana | 5 |
| Pruebas e Implantación | - Ejecución de pruebas unitarias e integradas- Corrección de errores- Despliegue en Firebase Hosting | 1 semana | 6 |

### Cronograma

| semana | 1 | 2 | 3 | 4 | 5 | 6 |
| --- | --- | --- | --- | --- | --- | --- |
| Auth/Roles | x |  |  |  |  |  |
| pedidos |  | x | x |  |  |  |
| Inventario |  |  |  | x |  |  |
| caja/pagos |  |  |  |  | x |  |
| pruebas/implantación |  |  |  |  |  | x |

## Planificación del Desarrollo – Cronograma de Módulos

Para organizar el desarrollo del sistema FastPOS y garantizar que cada módulo se construyera de manera ordenada y eficiente, se elaboró un **cronograma de desarrollo** asignando tiempos estimados para cada módulo clave:

| Módulo | Actividades principales | Duración estimada |
| --- | --- | --- |
| Autenticación y Roles | Configuración de Firebase Auth, creación de roles (mesero, caja, administrador), diseño de vistas de login y control de acceso | 1 semana |
| Gestión de Pedidos | Creación de interfaz de pedidos, registro en Firestore, cálculo automático de precios e impuestos, asociación con mesa/cliente | 2 semanas |
| Inventario | Modelado de datos de productos e insumos, actualización automática según pedidos, vistas de administración de stock | 1 semana |
| Pagos y Caja | Flujo de cobro, integración con pedidos, actualización de caja, generación de reportes básicos de ventas | 1 semana |

### Notas adicionales

* El cronograma asegura que el desarrollo siga un **orden lógico**: primero autenticación, luego gestión de pedidos, después inventario y finalmente pagos.
* Permite **controlar tiempos** y facilitar la toma de decisiones en caso de ajustes o retrasos.
* Sirve como evidencia de la competencia: **“Planifica proyectos informáticos, ofreciendo alternativas para la toma de decisiones de acuerdo a los requerimientos de la organización”**.

## Gestión de Incidencias – Corrección de Cálculo de Impuestos

**Situación detectada:** Durante las pruebas de integración del módulo de pedidos, se detectó un **error en el cálculo de impuestos (IVA)** en algunos pedidos con ingredientes adicionales y descuentos aplicados.

**Acción tomada:**

1. Se **asignó la tarea de corrección** al equipo de frontend, responsable de la lógica de cálculo de totales.
2. Se estableció un **plazo corto** para la corrección, asegurando que no se retrasara la entrega final del proyecto.
3. Se documentaron los cambios realizados en el código, incluyendo:
   * Ajuste en la función que suma ingredientes adicionales y aplica descuentos.
   * Revisión de la fórmula de cálculo de IVA.

**Resultado esperado y obtenido:**

* El sistema ahora calcula correctamente el **subtotal, impuestos y total final** en todos los casos.
* La entrega del proyecto se mantuvo según lo planificado.  
  Se valida que las modificaciones no afectaron otras funciones de la app (pruebas de regresión exitosas).

**Competencia asociada:**

* **Controla proyectos informáticos**, ofreciendo alternativas para la toma de decisiones de acuerdo a los requerimientos de la organización.

## Diseño de la colección Pedidos en Firestore

**Objetivo:** Crear un modelo de datos que permita registrar pedidos de manera eficiente, soportando consultas futuras por fecha, estado y cliente, y asegurando escalabilidad del sistema FastPOS.

**Colección:** Pedidos

**Campos y descripción:**

| Campo | Tipo | Descripción |
| --- | --- | --- |
| idPedido | string | Identificador único del pedido. |
| idCliente | string | Referencia al cliente que realiza el pedido (opcional si hay clientes registrados). |
| items | array | Lista de productos incluidos en el pedido, con detalle de cantidad y precio individual. |
| precioTotal | number | Suma de los precios de los productos y extras, antes de impuestos y descuentos. |
| estado | string | Estado del pedido: pendiente, en preparación, listo, entregado, cancelado. |
| fecha | timestamp | Fecha y hora en que se registró el pedido. |

**Consideraciones de diseño:**

1. Se diseñó para soportar **consultas por fecha** (por ejemplo: pedidos del día, semana o mes).
2. Se considera el **estado del pedido** para filtrar pedidos pendientes o entregados.
3. Escalable: permite agregar nuevos campos si en el futuro se incluyen promociones, notas especiales o métodos de pago.
4. Integración con otros módulos: el campo items se conecta con la colección Productos, y idCliente con la colección Usuarios.

**Competencia asociada:**

* **Diseña modelos de datos para soportar los requerimientos de la organización de acuerdo a un diseño definido y escalable en el tiempo.**

## Implementación de colecciones y reglas de seguridad en Firestore

**Objetivo:** Asegurar que los datos del sistema FastPOS estén correctamente almacenados y protegidos, permitiendo acceso solo a usuarios autenticados según su rol.

### Colecciones implementadas

| Colección | Descripción |
| --- | --- |
| Usuarios | Contiene información de los usuarios del sistema (nombre, correo, rol). |
| Roles | Define los permisos asociados a cada tipo de usuario: mesero, caja, administrador. |
| Productos | Lista de productos disponibles, precios y categorías. |
| Pedidos | Registro de pedidos realizados, incluyendo idCliente, items, precioTotal, estado y fecha. |

### Reglas de seguridad aplicadas

* Solo **usuarios autenticados** pueden leer o escribir datos en Firestore.
* Se diferencian permisos según rol:
  + **Mesero:** puede crear pedidos y leer productos.
  + **Caja:** puede ver todos los pedidos y calcular totales.
  + **Administrador:** puede administrar usuarios, productos, roles e inventario.
* Prevención de modificaciones no autorizadas mediante reglas de Firestore:

rules\_version = '2';

service cloud.firestore {

match /databases/{database}/documents {

match /Usuarios/{userId} {

allow read, write: if request.auth != null && request.auth.uid == userId;

}

match /Roles/{roleId} {

allow read: if request.auth != null;

allow write: if request.auth.token.admin == true;

}

match /Productos/{productId} {

allow read: if request.auth != null;

allow write: if request.auth.token.admin == true;

}

match /Pedidos/{pedidoId} {

allow create: if request.auth != null;

allow read: if request.auth != null;

allow update, delete: if request.auth.token.admin == true;

}

}

}

### Resultados esperados

* Los usuarios sólo acceden a lo que les corresponde según su rol.
* Datos sensibles de clientes y transacciones protegidos.
* La implementación soporta **escalabilidad futura**, pudiendo agregar nuevas colecciones y reglas sin afectar la seguridad existente.

**Competencia asociada:**

* **Implementa modelos de datos para soportar los requerimientos de la organización de acuerdo a un diseño definido y escalable en el tiempo.**

## Desarrollo del Frontend en React

**Objetivo:** Construir una interfaz de usuario intuitiva y modular para el sistema FastPOS, permitiendo la reutilización de componentes y facilitando el mantenimiento futuro.

### Componentes principales

| Componente | Funcionalidad |
| --- | --- |
| Login | Pantalla de autenticación de usuarios y control de roles (mesero, caja, administrador). |
| Mesa | Vista para seleccionar mesas y asociar pedidos a cada una. |
| Pedido | Gestión de pedidos: selección de productos, adición de ingredientes extra, cálculo de totales e impuestos. |
| Inventario | Gestión de productos disponibles, visualización de stock y actualización de cantidades. |

### Buenas prácticas aplicadas

1. **Separación de responsabilidades:** Cada componente maneja únicamente su lógica y UI, facilitando la comprensión y mantenimiento del código.
2. **Reutilización:**
   * Componentes como ProductoCard o ListaPedidos se usan en diferentes módulos (Pedido e Inventario).
   * Funciones utilitarias para cálculo de totales, impuestos y descuentos se centralizan en un archivo utils.js.
3. **Conexión con backend:**
   * Todos los componentes consumen datos desde Firestore mediante hooks personalizados (usePedidos, useProductos) para mantener consistencia y simplificar el manejo de estados.
4. **Escalabilidad:**
   * Se diseñó la estructura de carpetas por módulos (components, pages, hooks, utils) para facilitar la incorporación de nuevas funcionalidades en el futuro.

### Resultado esperado

* Interfaz clara y funcional para los distintos roles de usuario.
* Facilidad para agregar nuevos componentes o modificar existentes sin afectar el resto del sistema.
* Código mantenible y consistente con estándares de desarrollo modernos en React.

**Competencia asociada:**

* Construye una solución de software utilizando técnicas que permitan sistematizar el proceso de desarrollo y mantenimiento, asegurando el logro de los objetivos.

## Integración de módulos – Pedido, Inventario y Caja

**Objetivo:** Verificar que los distintos módulos del sistema FastPOS trabajen de manera integrada, garantizando consistencia en inventario y totales de caja al registrar un pedido.

### Flujo de integración

1. **Módulo Pedido**
   * **El mesero selecciona productos y agrega ingredientes adicionales.**
   * **Se calcula el subtotal del pedido en tiempo real.**
2. **Módulo Inventario**
   * **Cada vez que se agrega un producto al pedido, el inventario correspondiente se descuenta automáticamente.**
   * **Evita vender productos agotados y mantiene actualizados los niveles de stock.**
3. **Módulo Caja**
   * **El total del pedido, incluyendo impuestos y descuentos, se refleja automáticamente en el módulo Caja.**
   * **Permite procesar el pago correctamente y registrar la transacción en la base de datos.**

### Ejemplo práctico

| Acción | Resultado esperado |
| --- | --- |
| Mesero agrega 1 Pizza Mediana y 1 extra Pepperoni | Inventario: Pizza Mediana -1, Pepperoni -1; Total en Caja = subtotal + IVA |
| Mesero agrega otra Pizza Grande sin extras | Inventario actualizado automáticamente; Totales de Caja suman ambos pedidos |
| Se elimina un producto | Inventario se incrementa; Totales de Caja se recalculan automáticamente |

### Notas

* Este flujo asegura que la información sea consistente entre módulos.
* Permite detección temprana de errores en inventario o cálculo de totales.  
  Facilita la experiencia del usuario al garantizar que todo se actualice en tiempo real sin intervención manual.

### Competencia asociada:

* Integra los distintos componentes de una solución de software utilizando técnicas que permitan sistematizar el proceso de desarrollo y mantenimiento, asegurando el logro de los objetivos.

## Tabla Resumen de Evidencias por Rol – FastPOS

| Rol | Funcionalidad | Evidencia | Resultado esperado / comprobado |
| --- | --- | --- | --- |
| Mesero | Login y acceso al sistema | Inicia sesión con correo y contraseña | Acceso exitoso al sistema sin errores |
|  | Selección de mesa | Selecciona la mesa correspondiente | Mesa asignada correctamente al pedido |
|  | Creación de pedidos | Agrega productos con extras y confirma pedido | Subtotal y total calculados correctamente; pedido registrado en Firestore |
|  | Actualización de inventario | Confirma pedido | Inventario actualizado automáticamente (productos descontados) |
|  | Visualización del estado de pedidos | Consulta pedidos pendientes/en preparación | Estado de cada pedido actualizado en tiempo real |
| Caja | Login y acceso al sistema | Inicia sesión con credenciales válidas | Acceso exitoso al sistema sin errores |
|  | Visualización de todos los pedidos | Consulta pedidos de todos los meseros | Se muestran correctamente todos los pedidos activos |
|  | Cálculo de totales y cobro | Procesa pago de pedido | Total correcto (incluye impuestos y descuentos); caja actualizada |
|  | Reportes básicos | Genera reporte diario de ventas | Reporte generado correctamente y consistente con los pedidos |
| Administrador | Login y acceso al sistema | Inicia sesión con credenciales de admin | Acceso exitoso con permisos completos |
|  | Gestión de usuarios y roles | Crea, modifica o elimina usuarios | Cambios aplicados correctamente y reflejados en Firestore |
|  | Gestión de productos e inventario | Agrega, edita o elimina productos | Inventario actualizado automáticamente; cambios visibles en pedidos futuros |
|  | Visualización completa de pedidos | Filtra pedidos por estado o fecha | Todos los pedidos accesibles y correctamente filtrados |
|  | Revisión de logs y seguridad | Accede a reglas de seguridad de Firestore | Solo los roles autorizados pueden modificar o leer datos sensibles |

### Notas

1. Todas las pruebas se realizaron en Firebase Hosting ([https://fastpos-pizzeria.web.app](https://fastpos-pizzeria.web.app/)) y con datos en tiempo real de Firestore.
2. Cada funcionalidad fue verificada con múltiples combinaciones de roles y acciones simultáneas, asegurando consistencia y ausencia de errores.
3. Esta tabla sirve como evidencia directa del cumplimiento de la competencia:
   * Implanta una solución de software utilizando técnicas que permitan sistematizar el proceso de desarrollo y mantenimiento, asegurando el logro de los objetivos.