Proyecto BrokerTEC — Enunciado

CE-3101 Bases de Datos

MSc. Andrés Vargas — Escuela de Ingeniería en Computadores, TEC

II Semestre 2025

1. Objetivos del proyecto

Objetivo general

Evaluar y aplicar repositorios y arquitecturas de almacenamiento de datos para seleccionar e implementar la alternativa más adecuada que salvaguarde la información del proyecto **BrokerTEC**, de acuerdo con requisitos funcionales y no funcionales específicos.

Objetivos específicos

1. Interpretar los conceptos fundamentales de los SGBD para su uso correcto en el resguardo de datos (autenticación, transacciones y auditoría).
2. Identificar y diseñar el modelo de datos (desde ER hasta relacional) que cumpla los requerimientos del proyecto y garantice integridad y consistencia.
3. Examinar consideraciones de bases para asegurar consistencia en operaciones críticas (comprar, vender, liquidar).
4. Inspeccionar y justificar arquitecturas y mecanismos para manejar grandes volúmenes de datos (histórico de precios y reportes).

2. Historia y contexto

BrokerTEC nació en el **comedor del TEC**, un mediodía lluvioso, mientras tres estudiantes (María, Diego y Akemi) compartían la famosa **sopa azteca**. Entre cucharadas y bromas, concluyeron que los simuladores de bolsa eran o demasiado fantasiosos o inútiles para aprender bases de datos. "*Hagamos un juego serio, con reglas simples de verdad: todo en USD, precios que cambian en el tiempo, inventario real y roles separados*", dijo Akemi. Así surgió la idea: una plataforma donde **empresas públicas** se negocian en **mercados**, con **precio de acción** actual y **precios anteriores**, **cantidad de acciones** disponibles y **capitalización de mercado**, y donde las decisiones (comprar, vender, liquidar) quedan **trazadas** de principio a fin.

3. Descripción general del proyecto

BrokerTEC es un entorno educativo (todo en **USD**) para practicar decisiones de compra y venta de **acciones** de **empresas públicas** en uno o varios **mercados**. Cada empresa tiene: **precio de la acción** (actual y anteriores), **cantidad de acciones** y **capitalización de mercado** (precio actual $\times$ cantidad de acciones). El **inventario** lo administra una cuenta sistémica llamada **Tesorería**: si no hay acciones disponibles, no se compra. Un **Trader** opera con un **wallet** en USD y un **límite diario de recarga** según su **categoría** (junior/mid/senior). El **Admin** gobierna mercados, empresas, precios y usuarios (incluye **deshabilitar** y **delistar** con liquidación automática). El **Analista** solo observa reportes por **alias** (sin PII), gráficos **simples** y estadísticas de tenencia.

4. Resumen de acciones principales

1. **Operar (Trader)**: *Comprar/Vender* al **precio actual**; **Recargar** wallet respetando **límite diario**; **Liquidar todo** (confirmación de contraseña).
2. **Catálogos (Admin)**: **CRUD** de **Mercados** y **Empresas**; actualización de **precio actual** y registro de **precios anteriores** (histórico); gestión de **cantidad de acciones** y **capitalización**.
3. **Gobierno de usuarios (Admin)**: crear/asignar categoría y mercados; **deshabilitar** usuario con justificación; **delistar** empresa (ambos liquidan posiciones).
4. **Reportes (Analista)**: historial por **empresa** y por **usuario (alias)**, **inventario** de Tesorería y estadísticas de tenencia (traders vs. administración).
5. **Perfil personal (Todos)**: **CRUD** de nombre, alias, dirección, país de origen, teléfono, correo y cambio de contraseña.

5. Roles del sistema (visión general)

Admin: dueño de catálogos; carga/actualiza **precio actual** y **precios anteriores**; define límites por categoría; habilita mercados; **deshabilita** usuarios y **delista** empresas (liquidación). Ve **Top traders**.
Trader: opera su cuenta en USD; recarga (límite diario por categoría), compra/vende, consulta portafolio y puede **liquidar todo**. En portada ve **Top empresas** por capitalización en sus mercados.
Analista: solo lectura; ve reportes por **empresa** y por **alias**, **Precio vs. Tiempo** y estadísticas simples (traders vs. administración).

6. Diseño de UI por rol: vistas, datos y acciones

Nota de gráficos (simplicidad)

Sólo se permiten tres tipos de gráficos, y **únicamente** donde aporten claridad:

1. **Precio de la acción vs. tiempo** (línea simple).
2. **Top empresas por capitalización** (barras horizontales).
3. **Top traders** por dinero en wallet y por valor en acciones (barras horizontales).

6.1. Vistas comunes (todos los roles)

Perfil personal (CRUD)

1. **Objetivo:** permitir que cada usuario mantenga su propia información.
2. **Datos en pantalla:** nombre, **alias** (único), dirección, país de origen, teléfono, correo; sección para **cambio de contraseña**.
3. **Acciones:** crear/actualizar/eliminar campos personales; cambiar contraseña.
4. **Restricciones/validaciones:** alias único; correo válido; contraseña no visible (hash).
5. **Mensajes de error esperados:** alias duplicado, email inválido, contraseña débil.

6.2. Trader

Portada (Home)

1. **Objetivo:** dar un panorama rápido del mercado donde puede operar.
2. **Datos en pantalla:** por cada **mercado habilitado**, **Top 5 empresas por capitalización** (capitalización = precio actual × cantidad de acciones); **precio actual** y variación vs. *precio anterior* inmediato.
3. **Acciones:** navegar a **Empresa (detalle)** o directamente a **Operar**.
4. **Gráfico permitido: Top empresas** (barras horizontales).
5. **Restricciones/validaciones:** sólo mercados habilitados; datos en USD.
6. **Mensajes de error esperados:** "sin datos de capitalización" (si falta cantidad de acciones) o "mercado no habilitado".

Empresa (detalle)

1. **Objetivo:** entender la empresa antes de operar.
2. **Datos en pantalla: precio actual**, lista de **precios anteriores** (histórico simple), **cantidad de acciones** (totales), **acciones disponibles** (inventario Tesorería), **capitalización actual** y **mayor tenedor** por alias (si es Tesorería, mostrar "`administracion`").
3. **Acciones:** ir a **Operar** con la empresa precargada; (opcional) marcar como favorita.
4. **Gráfico permitido: Precio vs. tiempo** (línea).

5. **Restricciones/validaciones:** el **precio actual** es el último registro; histórico sólo *precios anteriores*.
6. **Mensajes de error esperados:** "sin histórico suficiente", "inventario no disponible".

Operar

1. **Objetivo:** comprar o vender de forma clara y segura.
2. **Datos en pantalla: precio actual**, campo **cantidad a comprar/vender**, **máximo comprable** = mín($\lfloor cash/precioactual \rfloor$, $accionesdisponibles$).
3. **Acciones: Comprar** / **Vender**.
4. **Gráfico permitido:** *Opcional*.
5. **Restricciones/validaciones:**
   - **Compra:** mercado habilitado, precio actual disponible, **inventario** suficiente en Tesorería y **cash** suficiente.
   - **Venta: posición suficiente**.
   - Operación **atómica** sobre wallet, posición e inventario; todo en USD.
6. **Mensajes de error esperados:** "saldo insuficiente", "inventario insuficiente", "posición insuficiente".

Wallet

1. **Objetivo:** gestionar el efectivo disponible.
2. **Datos en pantalla:** saldo, **categoría** (junior/mid/senior), **límite diario** y **consumo del día**; historial de recargas.
3. **Acciones: Recargar** hasta el límite restante.
4. **Gráfico permitido:** *Ninguno*.
5. **Restricciones/validaciones:** suma diaria de recargas $\leq$ límite; si excede, **rechazar**.
6. **Mensajes de error esperados:** "se alcanzó el límite diario", "monto inválido".

Portafolio

1. **Objetivo:** ver valor y composición de la cartera.
2. **Datos en pantalla:** por empresa: mercado, **cantidad**, **costo promedio**, **precio actual**, **valor actual**; y **totales** (cartera + wallet).
3. **Acciones:** comprar más de una empresa ya tenida; vender parcialmente (sin obligación de quedar en cero).
4. **Gráfico permitido:** *Ninguno*.
5. **Restricciones/validaciones:** sólo posiciones del propio Trader; valores en USD.

6. **Mensajes de error esperados:** “datos de precio no disponibles”.

Seguridad

1. **Objetivo:** acciones sensibles con confirmación.
2. **Datos en pantalla:** botón **Liquidar todo** (explicación y confirmación), último acceso.
3. **Acciones: Liquidar todo** (requiere **reingreso de contraseña**).
4. **Gráfico permitido:** *Ninguno.*
5. **Restricciones/validaciones:** vende **todas** las posiciones al **precio actual**; acción auditada; **irreversible**.
6. **Mensajes de error esperados:** “contraseña incorrecta”.

6.3. Admin

Catálogos (CRUD)

1. **Objetivo:** mantener datos maestros del sistema.
2. **Datos en pantalla: Mercados** (USD) y **Empresas** con **precio actual**, **precios anteriores**, **cantidad de acciones** y **capitalización**.
3. **Acciones: Crear/Actualizar/Eliminar** mercados y empresas; **Delistar** empresa.
4. **Gráfico permitido:** *Opcional.*
5. **Restricciones/validaciones:** delistar $\Rightarrow$ **liquidación automática** al **precio actual** (o fijado) y abono al wallet; registrar evento y justificación.
6. **Mensajes de error esperados:** “empresa con posiciones activas (se liquidarán)”, “datos de capitalización incompletos”.

Precios & Carga

1. **Objetivo:** cargar/validar **precio actual** y **precios anteriores**.
2. **Datos en pantalla:** listado de empresas con último precio y fecha/hora de actualización.
3. **Acciones:** carga **manual** y por **API** (auth básica ADMIN) de empresas, mercados y precios.
4. **Gráfico permitido: Precio vs. tiempo** para validación.
5. **Restricciones/validaciones:** precio > 0; timestamps válidos; contraseñas **cifradas** en BD; auditoría de carga.
6. **Mensajes de error esperados:** “precio inválido”, “auth fallida”, “formato de fecha inválido”.

Usuarios & Cuentas

1. **Objetivo:** gestionar el acceso y límites de operación.
2. **Datos en pantalla:** listado (alias, rol, estado), cuentas (wallet, **mercados habilitados**), **categoría** y **límite** (override posible).
3. **Acciones:** crear/editar usuarios; asignar categoría; habilitar mercados; **Deshabilitar** usuario (con **justificación**).
4. **Gráfico permitido: Top traders** (barras horizontales): *Top 5 por dinero en wallet* y *Top 5 por valor en acciones.*
5. **Restricciones/validaciones:** deshabilitar ⇒ **liquidar posiciones** del usuario al **precio actual**; el usuario queda **sólo lectura**; acción auditada.
6. **Mensajes de error esperados:** "justificación requerida", "usuario ya deshabilitado".

6.4. Analista

Empresa (reportes)

1. **Objetivo:** estudiar actividad y tenencia por empresa.
2. **Datos en pantalla: historial de transacciones** (alias, compra/venta, cantidad, precio, fecha–hora), **mayor tenedor** por alias (o "administracion"), **inventario** de Tesorería.
3. **Acciones:** filtros por fechas y por mercado.
4. **Gráfico permitido: Precio vs. tiempo** (línea simple).
5. **Restricciones/validaciones:** sólo **alias**, nunca PII.
6. **Mensajes de error esperados:** "rango de fechas inválido".

Usuario (alias)

1. **Objetivo:** revisar el comportamiento de un usuario (por alias).
2. **Datos en pantalla:** historial del alias (compras/ventas, cantidades, precios, fechas).
3. **Acciones:** ordenar/filtrar por fecha, empresa, tipo (compra/venta).
4. **Gráfico permitido:** *Ninguno.*
5. **Restricciones/validaciones:** sin PII.
6. **Mensajes de error esperados:** "alias inexistente".

Estadísticas (mercado)

1. **Objetivo:** ver distribución de tenencia por mercado/empresa.
2. **Datos en pantalla:** % de acciones en **traders** vs. **administración** por mercado y por empresa.
3. **Acciones:** cambiar nivel (empresa ↔ mercado).
4. **Gráfico permitido:** *Creatividad del estudiante* (Min: tabla porcentual simple).
5. **Restricciones/validaciones:** cálculos con posiciones vigentes.
6. **Mensajes de error esperados:** "sin posiciones para calcular".

7. Restricciones de negocio

1. **Moneda:** todo en **USD**.
2. **Top-up diario:** suma de recargas por cuenta y día $\leq$ límite por **categoría** (junior/mid/senior) u override; si excede, **rechazar**.
3. **Compra:** mercado habilitado; **precio actual** disponible; **acciones disponibles** en Tesorería; **cash suficiente**. Informar **máximo comprable** si no alcanza.
4. **Venta: posición suficiente**.
5. **Atomicidad:** comprar/vender actualiza **en una sola operación** wallet, posición e inventario de Tesorería.
6. **Delistar / Deshabilitar usuario:** ambos **liquidan** posiciones al **precio actual** (o fijado) y abonan al wallet; registrar evento y justificación (si aplica).
7. **Visibilidad y privacidad:** el Analista sólo ve **alias**; el Trader sólo su propia info; el Admin ve todo.
8. **Contraseñas:** siempre **cifradas**; nunca se exponen. *Liquidar todo* requiere **reingreso de contraseña**.
9. **Integridad de precios: precios anteriores** con tiempo válido y **precio actual** $>$ 0; "precio actual" es el último registro.

8. Descripción técnica del proyecto

8.1. Requerimientos técnicos (tecnologías)

1. **Backend:** Node.js (servidor de API REST).
2. **Frontend:** React (aplicación web).
3. **Base de datos:** Microsoft SQL Server.
4. **Gráficos:** librería de gráficos *open source* (línea y barras) para:

a) Precio de la acción vs. tiempo.

b) Top empresas por capitalización.

c) Top traders por dinero en wallet y por valor en acciones.

8.2. Ejecución local (obligatorio)

1. El proyecto debe **correr localmente** en una máquina estándar del estudiante.
2. Entregar **instrucciones claras** (README) para: levantar la base de datos, iniciar el backend y el frontend, y poblar datos de ejemplo.
3. Incluir **variables de entorno** (archivo de ejemplo) para credenciales y conexión a la BD.
4. Proveer **datos semilla** (seed) mínimos para evaluar flujos básicos.

8.3. Soporte móvil (obligatorio)

1. La solución debe ser **usable en dispositivos móviles**.
2. Puede lograrse mediante **diseño responsive** (la misma web se adapta a pantallas pequeñas) o mediante una **app móvil**.
3. La interfaz debe considerar **tacto**, **tamaños de fuente legibles** y **controles accesibles**.

8.4. Acceso a datos: ORM + SQL

1. Se **permite** usar un **ORM** en Node.js para **CRUD** y consultas simples.
2. Para **consultas complejas**, **reportes** y **operaciones críticas** (ej. comprar/vender, liquidar), se deben escribir **queries SQL** directamente sobre SQL Server.
3. Las operaciones que afecten dinero/posiciones deben ejecutarse en **transacciones** de base de datos.

8.5. Autenticación (requerido)

1. La autenticación de usuarios debe implementarse con usuario y contraseña , pueden investigar acerca de **JWT** (tokens) o hacerlo con su implementación siguiendo lineamientos de seguridad.
2. Proteger endpoints según **rol** (Admin, Trader, Analista).

8.6. Documentación de API (puntos extra)

1. **Swagger/OpenAPI** es un **estándar** para describir APIs (rutas, parámetros, respuestas y errores) en un formato legible por humanos y máquinas.

2. Beneficios:

   a) **Contrato claro** entre frontend y backend.

   b) **Pruebas** y exploración de endpoints desde un navegador.

   c) **Trazabilidad** de cambios y **consistencia** en respuestas.

3. **Puntos extra:** publicar la documentación de su API en una ruta dedicada (*por ejemplo*, `/docs`) y mantenerla actualizada.

8.7. Recomendaciones técnicas

Hash de contraseñas.

- **Nunca** almacene contraseñas en texto plano.
- Almacene **sólo** un **hash** seguro (con **sal** aleatoria). Un hash es una *huella irreversible* de la contraseña: permite verificar sin conocer la original.

Auditoría.

- Mantener un registro de **quién** hizo **qué**, **cuándo** y **sobre qué**: por ejemplo, delistar empresas, deshabilitar usuarios, *liquidar todo*, cambiar límites de recarga o cargar precios.
- La auditoría facilita **trazabilidad**, **revisión de incidentes** y **evaluación** del proyecto.

RBAC (Role-Based Access Control).

- Control de acceso por **rol**: **Admin**, **Trader**, **Analista**.
- Asigne permisos según el **principio de mínimo privilegio** (cada rol sólo lo necesario).

Calidad y robustez.

- **Validación** de entradas (tipos, rangos, formatos) y **mensajes de error** claros.
- **Manejo de errores** consistente (códigos y estructura de respuesta).
- **Logs** legibles para depurar (sin imprimir datos sensibles).

8.8. Entregables técnicos mínimos

1. **README** con pasos para ejecutar localmente (BD, backend, frontend) y datos de ejemplo.
2. **Archivo de variables de entorno** de ejemplo (sin credenciales reales).
3. **Scripts** de inicialización/población de datos mínimos.
4. **Evidencias** de que funciona en **móvil** (capturas o demo).

9. Repositorio y control de versiones (obligatorio)

9.1. Repositorio en GitHub

1. Cada equipo debe alojar el código en un **repositorio de GitHub** (uno por proyecto).
2. **Agregar como colaborador** al profesor: **afelipe.vargas.r@gmail.com**.
3. **Visibilidad:** puede ser privado o público. Si es privado, asegúrense de que el profesor tenga acceso.
4. **Rama por defecto:** `main`.
5. **Buenas prácticas mínimas:** README con instrucciones de ejecución local, archivo de variables de entorno de ejemplo, datos semilla, y carpeta `/docs` (si aplicara).

Cómo agregar al profesor como colaborador (UI de GitHub)

1. Ir a *Settings* del repositorio → *Collaborators*.
2. Clic en *Add people* y escribir: `afelipe.vargas.r@gmail.com`.
3. Asignar permiso al menos de **lectura** (read).
4. Confirmar invitación.

9.2. Release branch y tag para la entrega

¿Qué es una *release branch*? Es una **rama temporal** para **congelar** el estado del proyecto de cara a una entrega. En la release branch **sólo** se aceptan **correcciones (bugfixes)** y ajustes menores; no se agregan features nuevas.

¿Qué es un *tag* de Git? Es una **etiqueta inmutable** que marca un **commit específico** (por ejemplo, la versión entregada). Conviene usar **versionado semántico**: `v1.0.0`, `v1.1.0`, `v1.1.1`, etc.

Flujo requerido para la fecha de entrega

1. **Crear la release branch** desde `main` (o desde `develop` si usan ese flujo):

```
git checkout main
git pull
git checkout -b release/1.0.0
git push -u origin release/1.0.0
```

2. **Crear un tag de versión** apuntando al **commit** final de `main`:

```
git checkout main
git pull
git tag -a v1.0.0 -m "Release 1.0.0"
git push origin v1.0.0
```

3. **(Recomendado)** Crear un **GitHub Release** usando ese tag (*Releases → Draft a new release*), adjuntando:
   - Instrucciones de ejecución rápida (1–2 párrafos).

9.3. Requisitos de entrega (control de versiones)

1. Repositorio en GitHub con el profesor agregado: `afelipe.vargas.r@gmail.com`.
2. Release branch creada para la entrega (por ejemplo, `release/1.0.0`).
3. Tag de versión publicado (por ejemplo, `v1.0.0`) **apuntando al commit final** entregado.
4. (Recomendado) GitHub Release creado a partir del tag, con changelog y artefactos.

graphicx pdfpages tabularx

10. Documentación del proyecto (entregables y estructura)

10.1. Entregables (obligatorios)

1. **Diagrama del modelo relacional** en **PDF** (exportado desde su herramienta). Debe mostrar:
   a) Entidades/tablas con **PK**, **FK**, **atributos** y **restricciones** clave.
   b) Relaciones representadas mediante claves foráneas y notas de **cardinalidad/participación**.
2. **Documento en LaTeX** (*este documento*): incluye **introducción**, **objetivos del documento**, **modelo relacional (PDF)**, **descripción de cada relación**, **tipos de datos usados**, **conclusión** y **recomendaciones del aplicativo**.
3. **Sección de formación AF1–AF4** (administración de proyectos y finanzas) con respuestas **desde la perspectiva del estudiante**.

10.2. Estructura mínima del documento (LaTeX)

1) Introducción

- Contexto breve del problema (retomar la historia "sopa azteca" y el objetivo del MVP).
- Alcance del modelo relacional presentado (qué cubre y qué no).

2) Objetivos de *este* documento

- Mismos objetivos del enunciado.

3) Explicación general de la solución

- **Qué muestra el diagrama relacional:** entidades principales (Usuarios/Cuentas, Empresas/Precios, Posiciones/Trades, Tesorería/Auditoría), claves primarias y foráneas, y las cardinalidades básicas (1:N) que conectan el flujo de datos.
- **Decisiones de datos (tipos y restricciones):** precios y montos como valores numéricos de precisión, cantidades de acciones como enteros no negativos, fechas/horas para histórico y trazabilidad, textos para alias/nombres, y estados/roles controlados para la lógica.
- **Reglas de negocio reflejadas:** comprar/vender actualiza coherentemente wallet, posición e inventario; límite diario de recarga por categoría; no se compra sin inventario ni se vende sin posición; delistar/deshabilitar liquida al precio actual.
- **Consultas y uso esperado:** precio de la acción vs. tiempo, top de empresas por capitalización y top de traders, portafolio e historial por alias/empresa; cómo estas consultas validan que el modelo cumple el MVP.

4) Conclusión y recomendaciones del aplicativo

- **Conclusión:** valor del diseño, qué aprendizajes deja, y si cumple los requisitos del MVP.
- **Recomendaciones:** mejoras futuras (índices adicionales, vistas/materializaciones para reportes, optimizaciones en consultas complejas, endurecer validaciones, etc.).

10.3. Sección formativa: Administración de proyectos y finanzas (AF)

Desde su perspectiva de estudiantes, completen la descripción y respondan la pregunta guía para cada atributo.

AF1 — Ciclo de vida del proyecto. Descripción (estudiante): Explique cómo identificaron fases (inicio, planificación, ejecución, cierre) y entregables clave.
Pregunta guía: ¿Qué hitos y criterios de aceptación definimos para pasar de una fase a la siguiente en BrokerTEC?

AF2 — Beneficios, costos económicos y financieros. Descripción (estudiante): Identifique beneficios esperados (aprendizaje, reuso, mantenibilidad) y costos (tiempo, servidores, esfuerzo de pruebas).
Pregunta guía: ¿Cómo justificamos nuestras decisiones de diseño (por ejemplo, tipos de datos/índices) en términos de costo–beneficio?

AF3 — Planificación financiera y entornos multidisciplinarios. Descripción (estudiante): Describa el plan para recursos (personas, tiempo, herramientas) considerando desarrollo, datos y UI.
Pregunta guía: ¿Cómo distribuimos tareas y riesgos entre roles (backend, datos, frontend) para cumplir plazos y objetivos?

AF4 — Herramientas de gestión y finanzas. Descripción (estudiante): Señale herramientas usadas (tableros, issues, control de versiones, métricas) y cómo ayudaron a lograr metas.
Pregunta guía: ¿Qué métricas o evidencias muestran que alcanzamos los objetivos (calidad, rendimiento, completitud de requisitos)?

10.4. Checklist de verificación (antes de entregar)

1. Diagrama relacional en PDF embebido y legible (PK, FK, tipos, restricciones).
2. Tabla(s) de **entidades** completa(s) y tabla(s) de **relaciones** con justificación de FK.
3. **Introducción**, **objetivos del documento**, **explicación de la solución**, **conclusión** y **recomendaciones** presentes.
4. Respuestas AF1–AF4 completas (descripción + pregunta guía).

11. Cronograma y entrega

- **Medio de entrega:** TECDigital.
- **Fecha y hora límite: Martes 21/10/2025, 11:59 p.m.**
- **Valor del proyecto: 25 %** de la nota del curso.
- **Cita:** Definir con el profesor en clase.

12. Rúbrica de calificación (100 %)

Resumen de pesos

Rubro	Peso	Evidencia principal
A. Funcionalidad general (Desktop)	12 %	App corre local, flujo básico en pantalla grande, contr
A2. Soporte móvil (Responsive/App)	8 %	Usabilidad en móvil, vistas clave operativas
B. Funcionalidad por rol: Trader	20 %	Acciones simples operativas
C. Funcionalidad por rol: Admin	15 %	Acciones simples operativas
D. Funcionalidad por rol: Analista	10 %	Acciones simples operativas
E. Documentación (en LaTeX)	15 %	Documento completo y coherente
F. Diagrama del modelo relacional (PDF)	10 %	PK, FK, cardinalidad/participación
G. Preguntas de atributos (AF1–AF4)	5 %	Respuestas del estudiante
H. Defensa	5 %	Demo, claridad y Q&A

Criterios detallados por rubro

A. Funcionalidad general (Desktop) — 12 %.

- **Ejecución local:** backend, frontend y BD levantan con README claro y datos semilla.
- **Flujos básicos en desktop:** navegación, login, vistas principales (Home, Empresa, Operar, Wallet/Portafolio, Admin/Analista) *sin romper layout.*
- **Control de versiones:** repositorio en GitHub con el profesor agregado (`afelipe.vargas.r@gmail.c` **release branch** creada para la entrega y **tag** publicado que apunta al commit final.
- **Calidad básica:** validaciones, manejo de errores y mensajes claros; sin datos sensibles en logs.

A2. Soporte móvil (Responsive/App) — 8 %.

- **Usabilidad en móvil:** interfaz usable en pantallas pequeñas (360–414px), navegación táctil fluida.
- **Vistas clave operativas:** al menos Home, Empresa (detalle), Operar, Wallet/Portafolio *funcionan* en móvil (formularios, botones, tablas).
- **Legibilidad y accesibilidad:** tamaños de fuente adecuados, controles táctiles alcanzables, sin desbordes/scrolls horizontales indebidos.
- **Evidencia:** capturas o demo que muestren los flujos móviles funcionando.

B. Funcionalidad por rol: Trader — 20 %. *Acciones simples que deben funcionar de extremo a extremo:*

1. **Operar:** Comprar/Vender al **precio actual**, mostrando **máximo comprable**; operación atómica (actualiza wallet, posición, inventario).
2. **Wallet:** Recarga respetando **límite diario** por categoría; rechazo con mensaje si excede.
3. **Portafolio:** Visualiza cantidad, costo promedio, precio actual y valor; permite vender parcial.
4. **Seguridad: Liquidar todo** con reingreso de contraseña y efecto correcto en datos.
5. **Perfil (CRUD propio):** edición de datos personales y cambio de contraseña.

C. Funcionalidad por rol: Admin — 15 %.

1. **Catálogos (CRUD):** crear/editar/eliminar **Mercados** y **Empresas** (precio actual, precios anteriores, cantidad de acciones).
2. **Precios & Carga:** registrar precio actual y anteriores; visible última actualización.
3. **Usuarios & Cuentas:** asignar categoría, habilitar mercados, **deshabilitar** usuario (requiere justificación).

4. **Eventos críticos: Delistar** empresa y **deshabilitar** usuario $\Rightarrow$ $liquidacinautomticaalprecioactual, audito$

D. Funcionalidad por rol: Analista — 10 %.

1. **Reportes por empresa:** historial (alias, compra/venta, cantidad, precio, fecha–hora); mayor tenedor (alias o "administracion").
2. **Reportes por usuario (alias):** historial completo ordenable/filtrable (sin PII).
3. **Estadísticas:** distribución traders vs. administración por empresa/mercado (tabla simple).

E. Documentación en LaTeX — 15 %.

- **Estructura mínima:** Introducción, Objetivos del documento, **Modelo relacional (PDF embebido)**, Descripción de relaciones, Tipos de datos utilizados, Conclusión y Recomendaciones.
- **Claridad y coherencia:** términos del glosario bien aplicados; referencias cruzadas correctas.
- **Instrucciones técnicas:** README y guía de ejecución local consistentes con lo documentado.

F. Diagrama del modelo relacional (10 %).

- **Calidad del diagrama:** tablas con **PK/FK**, cardinalidades 1:N, atributos clave y notas de restricción.
- **Legibilidad:** tamaños, distribución y export en PDF nítido.

G. Preguntas de atributos (AF1–AF4) — 5 %.

- Respuestas **desde la perspectiva del estudiante** a AF1–AF4 (administración de proyectos y finanzas), conectadas al proyecto.
- Cada respuesta incluye **descripción breve** y **respuesta a la pregunta guía** definida en el enunciado.

H. Defensa — 5 %.

- **Demo funcional** y narrativa clara (qué, por qué, cómo).
- **Coherencia** entre demo, documentación y diagrama.
- **Q&A:** responde con precisión y evidencia (capturas/logs/queries).

Escala de desempeño (por rubro)

- **Excelente (100 % del rubro):** cumple todo lo indicado, sin fallos, con mensajes/validaciones claras.
- **Bueno (80 %):** cumple la mayoría; fallos menores no afectan flujos principales.
- **Básico (60 %):** cumple lo mínimo; hay ausencias parciales o incoherencias puntuales.
- **Insuficiente (0–40 %):** faltan elementos clave o los flujos principales no funcionan.

13. Glosario de términos

- **Acción:** unidad de propiedad de una empresa pública; cada acción representa una fracción del valor de la empresa.
- **Empresa pública:** compañía cuyas acciones pueden negociarse en un mercado.
- **Mercado:** lugar (bolsa) donde se compran y venden acciones; en el MVP, todo opera en **USD**.
- **Precio actual:** último *precio de la acción* registrado para una empresa.
- **Precios anteriores (histórico):** registros previos de *precio de la acción* a lo largo del tiempo.
- **Cantidad de acciones:** número de acciones de la empresa *totales*; **acciones disponibles** se refiere al inventario que administra Tesorería.

- **Capitalización de mercado:** *precio actual* × *cantidad de acciones*; aproxima el valor de mercado de la empresa.
- **Tesorería:** cuenta sistémica que mantiene el inventario de acciones "sin dueño"; compras restan inventario, ventas lo devuelven.
- **Wallet:** saldo en USD de la cuenta del usuario para operar; se alimenta con *recargas* (top-ups).
- **Top-up (recarga):** aumento del wallet; limitado por día según **categoría** del Trader.
- **Delistar:** retirar una empresa del sistema de negociación; se liquidan posiciones al **precio actual** y se abona al wallet.